MOSQUITO IDENTIFICATION USING INFRARED SPECTROSCOPY AND CHEMOMETRICS

A thesis presented to the faculty of the Graduate School of Western Carolina University in partial fulfillment of the requirements for the degree of Masters of Science in Chemistry.

By

Lamyae Sroute

Advisor: Dr. Scott Huffman
Associate Professor of Chemistry
Department of Chemistry & Physics

Committee Members: Dr. Brian Byrd, Environmental Health Sciences Program
Dr. Carmen Huffman, Department of Chemistry & Physics

April 2018

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Ae.	*Aedes*
A	spectral absorbance data
ANN	artificial neural network
ATR	attenuated total reflectance
c	cropping
d	second derivative
DRIFTS	diffused reflectance infrared fourier transform spectroscopy
FN	false negative
FP	false positive
FT-IR	fourier transform infrared
IR	infrared
JSON	JavaScript objective notation
m	mean centering
MALDI-TOF MS	matrix assisted laser desorption/ionization time-of-flight mass spectrometry
MCC	mathew's correlation coefficient
n	normalization
NIRS	near-infrared spectroscopy
NOLA	New Orleans, LA
PCR	polymerase chain reaction
PLS-DA	partial least squares discriminant analysis
PLS-R	partial least squares regression
PP	pre-processing
SG	Savitzky-Golay
SVM	support vector machine
T	all classified data set
TN	true negative
TP	true positive

ABSTRACT

MOSQUITO IDENTIFICATION USING INFRARED SPECTROSCOPY AND CHEMOMETRICS

Lamyae Sroute, Masters of Science in Chemistry

Western Carolina University (April 2018)

Advisor: Dr. Scott Huffman

Mosquito control interventions are more effective when informed by routine entomologic surveillance. Thus, accurate and rapid species identification remains a critical component of operational mosquito control. Current methods to identify adult mosquitoes rely chiefly on microscopic identification by trained personnel. In some larger mosquito control programs, molecular methods may be used for species or pathogen identification and advanced techniques (e.g., age-grading by ovarian dissection) may be used to further assess the mosquito population structure. Each of these methods are labor intensive and subject to a series of operator or laboratory errors. Therefore, there is a need for rapid and non-destructive species identification techniques that can be used on a scale that is ecologically, economically, and epidemiologically meaningful. Our current research aims to develop methods of biochemical discrimination between different mosquito species using infrared spectroscopy. Infrared spectroscopy is a sensitive, information rich technique that is capable of detecting a wide range of molecular signals ranging from subtle changes in protein secondary structure to transmembrane protein-lipid interactions. The resulting spectral data, when coupled with a numerical analysis (chemometrics) method, such as partial least squares discriminant analysis, may be used to classify mosquitoes by species or physiologic status. Herein, we have applied Fourier transform infrared (FT-IR) microspectroscopy to identify four container-inhabiting *Aedes* species (*Ae. aegypti*, *Ae. albopictus*, *Ae. japonicus*, and *Ae. triseriatus*) obtained from both field and laboratory conditions. At present, our FT-IR classification

success rate using partial least squares discriminant analysis, when compared to identification by a trained entomologist, is 94.5-100%. This method, which is rapid and easy to use, has the potential to decrease the labor costs and time associated with mosquito species identification. Further development coupled with process automation may provide operationally useful methods for rapid identification of many mosquito species and their physiologic status.

CHAPTER ONE: BACKGROUND

1.1 Importance of Mosquito Surveillance

According to the World Health Organization, mosquitoes infect over 300 million people a year. Of those infected, over 800 thousand die each year.[1] In the United States, mosquito-borne viruses such as West Nile, La Crosse and Eastern equine encephalitis viruses are transmitted to humans on an annual basis. At present, there are no effective human vaccines to protect against these mosquito-borne viruses. Thus, preventing and controlling the transmission of mosquito-borne transmitted disease to humans depends greatly on the application of mosquito control informed by surveillance. In 2017, American Mosquito Control Association (AMCA) released a manual called the *Best Practices for Integrated Mosquito Management*, where they stated that the identification of problem mosquito species is "the first step towards defining and developing control methods".[2] Mosquito identification has been one of the most effective and successful approached to understanding the population dynamics and species distribution of mosquitoes in a given area.[3,4] Constant surveillance is important for determining when transmission of disease risk is high and how to control mosquitoes. Population surveillance is especially important for disease vectoring mosquitoes such as *Aedes* container inhabiting species: *Ae. triseriatus*, *Ae. albopictus*, *Ae. japonicus* and *Ae. aegypti*. These mosquito species are capable of transmitting Zika,[5,6] dengue,[7] West Nile,[8] La Crosse,[9] Chikungunya[10] or other arboviruses.

1.2 Current Methods for Mosquito Speciation

Mosquito surveillance is generally performed by local mosquito control agencies that employ trained personnel, typically a competent biologist or entomologist, to perform routine microscopic identification of adult female mosquitoes caught by surveillance traps. Hiring trained personnel for routine mosquito identification comes with a large expense which some smaller control agencies cannot afford. An ill-equipped mosquito control program may result in unnecessary insecticide application which then increases the likelihood of mosquitoes developing in-

secticide resistance. Microscopic identification heavily relies on the identification of mosquito species through morphological distinguishing features, however, these features can sometimes be damaged during collection, making it difficult and sometimes impossible to identify field collected mosquitoes. In some larger mosquito control programs, DNA profiling through standard polymerase chain reaction (PCR) assays may be used to identify cryptic species (morphologically indistinguishable species). This method is both labor intensive and time consuming. Thus, there is a need for rapid and non-destructive species identification techniques that can be used on a scale that is ecologically, economically, and epidemiologically meaningful.

1.3 Alternative Methods

To overcome the drawbacks of classical microscopic identification methods and the PCR-based methods, alternative methods have been explored, notably whole cell matrix assisted laser desorption/ionization time-of-flight mass spectrometry (MALDI-TOF MS) and near-infrared spectroscopy (NIRS).[11,12] The MALDI-TOF MS technique was used as a molecular protein profiling tool for the identification of taxonomically closely related mosquito species.[11] Samples were prepared by homogenizing a mosquito head and thoraces in a Eppendorf tube containing formic acid, and a matrix suspension was developed in a separate tube using sinapic acid.[11] In contrast, NIRS was used to measure the absorbance of near-infrared light by a sample, the resulting spectral data are explored for characteristic markers that distinguish mosquito species.[12,13]

Although MALDI-TOF could potentially be a valuable method in mosquito identification of closely related mosquito species, the sample preparation involved is labor intensive and requires additional cost for consumables, making the process time-consuming and overall costly in comparison to other techniques. The NIRS method for identification also has its own drawbacks. For instance, the interpretation of NIRS is difficult due to the broad bands which are the result of the overlap spectral compression of many overtone and combination bands.[14] Also, NIRS has a much lower sensitivity.[14] Thus, the variations occurring between NIR spectra of different samples are very small, resulting in an overall lower selectivity compared to Fourier transform infrared (FT-

IR) spectroscopy .

1.4 Theory of Infrared Spectroscopy

Infrared spectroscopy utilizes the infrared region, 14000 cm^{-1} to 10 cm^{-1}, of the electromagnetic spectrum. The infrared portion of the electromagnetic spectrum is divided into three regions: near-infrared region (14000-4000 cm^{-1}), mid-infrared region (4000-400 cm^{-1}) and far-infrared region (400-10 cm^{-1}). The near-infrared radiation can excite overtone, and combination vibrations; the mid-infrared energy can be used to study fundamental vibrations of structures and the far-infrared region can be used to study large scale vibrations and vibrations of heavy atoms. The mid-infrared region is the most commonly used region for analysis due to nearly all molecules having characteristic absorbance wavenumbers and primary molecular vibrations in this region.[15]

Infrared spectroscopic methods involve the study of the interaction of infrared light with matter. As IR light interacts with specific groups of atoms in molecules, unique wavelengths are absorbed, exciting the groups of atoms to higher energy vibrational states. For infrared absorption to occur there must be a net change in dipole moment in a molecule as it vibrates. The number of vibrations (n) in a given molecule are known as modes and can be calculated using the following equation for nonlinear molecules

$$n = 3N - 6 \tag{1}$$

where N represents the number of atoms in the structure. During absorption, a vibrational transition from the group state (v = 0) to the first excited state (v = 1) occurs. The gap between each energy level (ΔE) can correspond to the frequency of light that excited the molecule.

$$\Delta E = hc\tilde{\nu} \tag{2}$$

where h is Planck′s constant, c is the speed of light and $\tilde{\nu}$ is wavenumber.

When a molecule vibrates, there is a fluctuation in its dipole moment, causing a field that in-

teracts with the electric field associated with the infrared light. A typical infrared spectrum consists of bands whose wavenumbers and intensities are specific to a given molecule. The wavenumber of each vibrational mode is governed by the reduced mass (μ) of the atoms involved in the vibration as well as the strength of the bonds (k) involved in the vibration, as described in the following equation.

$$v = \frac{1}{2\pi c}\sqrt{\frac{k}{\mu}} \tag{3}$$

Based on the equation, larger atoms with weak chemical bonds would be expected to have bands in the lower energy region of the infrared spectrum, while the opposite would be true for smaller atoms with stronger chemical bonds. Thus, there is a direct correspondence between infrared band positions and chemical structures in the molecule.

1.5 Infrared Spectroscopy Sampling Techniques

FT-IR spectroscopy is an extremely reliable and well recognized chemical fingerprinting technique which has the ability of obtaining spectra from a wide range of solids, liquids and gases. However, sample handling techniques are a vital part of developing a method which results in the best spectral data possible which in return will provide more consistent results. For infrared spectroscopy, there are several options for sampling techniques: transmission, attenuated total reflectance (ATR), specular reflectance and diffuse reflectance.

In figure 1 (a) is a schematic of the transmission geometry. Simply, the IR beam passes through the sample, the transmitted energy is measured and a spectrum is generated. The main advantages of transmission FT-IR are its high signal-to-noise ratios and its historical prevalence. However, this technique requires expertise in sample preparation that tends to be time consuming and unreproducible.[16]

Attenuated total reflection (ATR) is a surface sampling technique commonly used in conjunction with infrared spectroscopy for solid and liquid state samples. This technique allows samples

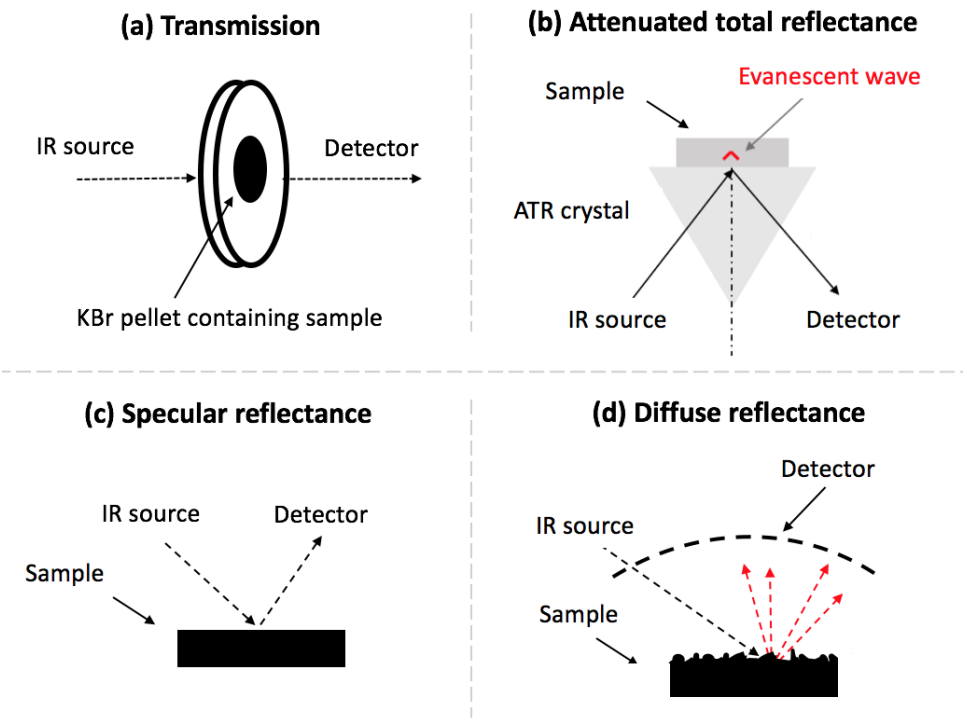

Figure 1. Schematics of infrared spectroscopy sampling techniques.

to be examined directly without a requirement for further preparation. A schematic of this technique is shown in Figure 1 (b). An infrared beam is directed onto an optically dense crystal with a high refractive index, typically a diamond, at a certain angle. The technique utilizes the property of total internal reflection, which results in an evanescent wave capable of penetrating a sample between 0.5 and 2 μm. The main advantage of the ATR technique is its ease of sample acquisition, in most cases not requiring any sample preparation, and near-universal applicability.[16,17] However, this technique can sometimes be destructive to sample due to the pressure applied and the presence of air between the sample and the ATR crystal can affect spectral data.[15]

Specular reflection is defined as light reflected from a smooth (mirror-like) surface, in that the angle of the incident light is exactly the same as the angle of reflected light, but on the opposite side of the surface, as shown in Figure 1 (c). This technique provides excellent qualitative data and is commonly used for surface characterization of thin layer polymers or coating on polished

metals.[16] However, for true specular reflection to occur, the sample must be large, flat and have a reflective surface.

Diffusive reflectance (DRIFTS) is another commonly used reflection measurement technique. DRIFTS is produced by rough surfaces, where the reflected light does not equal the angle of incidence as show in Figure 1 (d). The main advantages of DRIFTS is that it requires little to no sample preparation and has far more occurrences than specular reflection in everyday environment.[16,18]

1.6 Fourier Transform Infrared Microspectroscopy

Although many of the sampling methods discussed allow for chemical identification, the combination of them with microscopy (microspectroscopy) permits the examination of complex, spatially heterogeneous samples. Figure 2 illustrates reflection mode using an infrared microspectrometer, where the sample is placed on an inexpensive IR-reflecting surface (usually an aluminum sheet) and measurements are generated by an IR light directed to the surface and returned to the microscope objective by either specular or diffused reflection. The intensities of the recorded spectrum for a sample not only account for the chemistry of the sample but also the topographical features of the sample.[15]

1.7 Fourier Transform Infrared Spectroscopy for Classification

FT-IR spectroscopy is a type of vibrational spectroscopy method capable of producing an infrared spectrum that is sensitive to a wide range of molecular signals ranging from subtle changes in protein secondary structure to transmembrane protein-lipid interactions.[19–23] The resulting spectral features may be used to distinguish and identify various biological samples. Further multivariate data analysis allows the most significant spectral data to be extrapolated from high-dimensional spectral data, thus providing a fast and conventional technique for classification purposes.

Characterization and identification of microorganisms (i.e. bacteria) viruses and subtypes of human cancer using mid-infrared spectroscopy is well established.[24,25] The use of vibrational

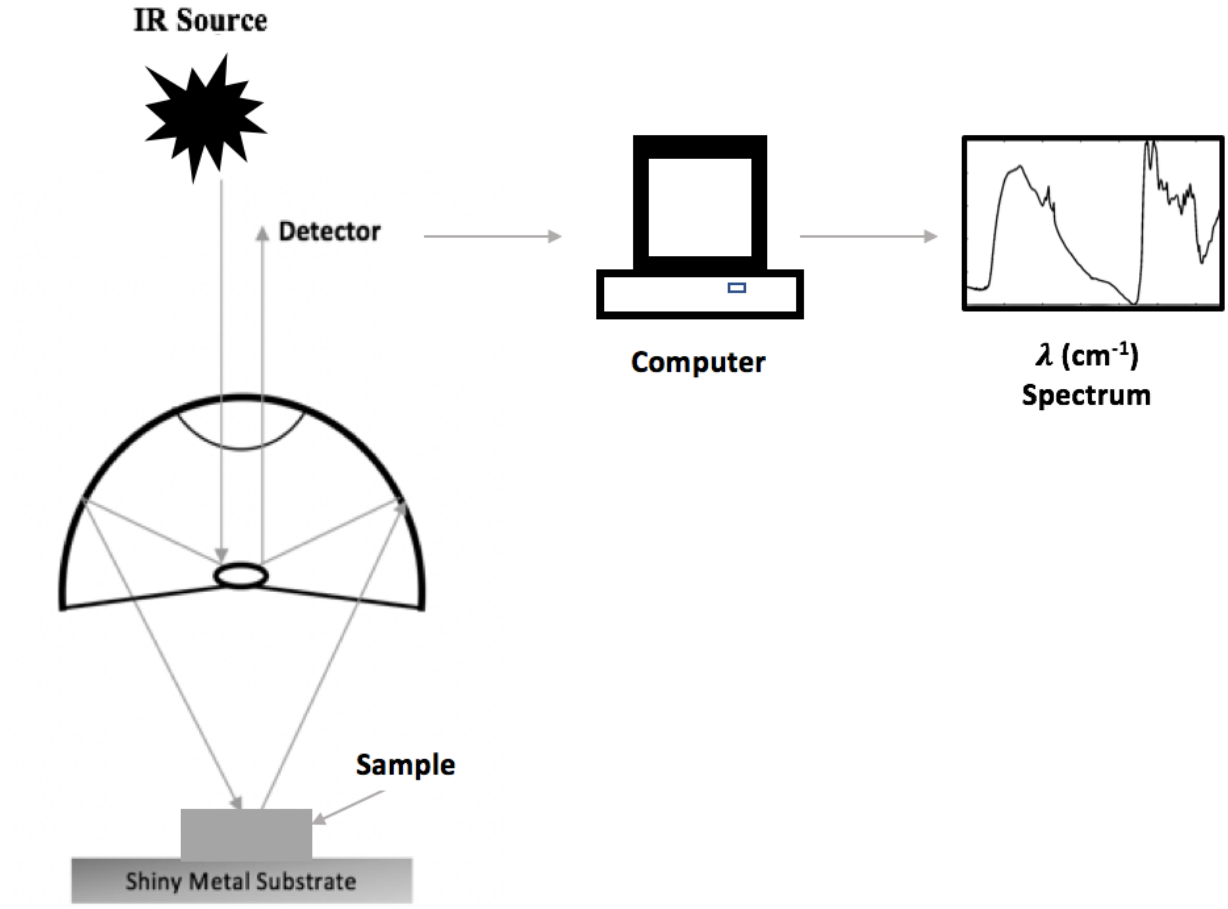

Figure 2. General schematics of an infrared microscope spectrometer.

spectroscopy for bacterial identification and classification has been evaluated for different species, particularly foodborne pathogens.[21,24] Vibrational spectroscopy has also been implemented as a bio-analytical tool for routine classification of normal and pathological tissue. This method has been used for various types of cancers including breast, endometrial, cervical, prostatic and brain cancers.[20]

Although the FT-IR spectroscopy is a well-developed, information-rich technique, there has been little application of this technique for classification in the mosquito field. In 2017, FT-IR based diagnostics were successfully implemented for detecting *Wolbachia* infection and identifying sex and distinguishing the chronologic age (2 versus 10 days) of *Aedes aegypti* mosquitoes

in Australia.[19] However, to our knowledge FT-IR spectroscopy has never been applied to species identification of mosquitoes.

1.8 Data Processing

Data processing for the purpose of classification is carried out in a sequence of steps depicted in Figure 3. Essentially the FT-IR spectral data is split into two data sets: training and validation. This is normally done by randomly splitting data in half, and serves the purpose of cross validating the model's performance. The training data set is used to build a classification model which is used to predict the classification of the validation data set. The purpose of model validation is to establish the uncertainty in the models predictive ability. Here we describe the following data analysis steps: data pre-processing, classifier training, and classification.

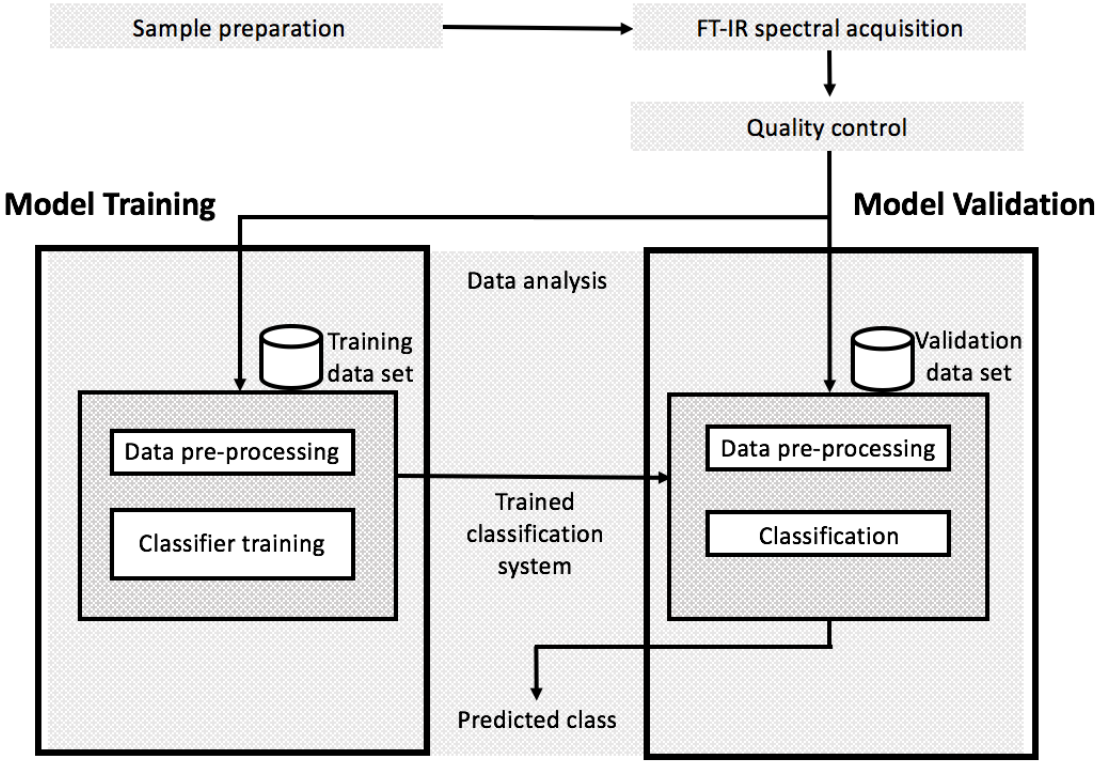

Figure 3. FT-IR spectroscopy work flow for classification.

1.8.1 Data Pre-processing

The purpose of data pre-processing spectral data is to improve the robustness and accuracy of subsequent multivariate analysis as well as correcting issues associate with data acquisition. The most common issues that arise with infrared spectroscopy of biological samples is sloped or oscillatory baselines as a result of scattering, high frequency instrumental noise, interference from compounds such as water vapor and carbon dioxide, and variation in sample thickness.[15] Thus, data pre-processing may improve classification by minimizing the presence of any confounding factors. The most common data pre-processing technique to resolve or minimize these effects are normalization, Savitzky-Golay (SG) smoothing and derivation, cropping and mean centering.

Normalizing

The goal of normalization is to eliminate the variations in intensity due to confounding factors such as varying thickness of samples and to correct for day-to-day variations such as humidity or pressure. Each spectrum is intensity normalized to Amide I band to scale the intensity values for all the spectra in the model so that their minimum is at zero and the Amide I band peak for all spectra are all the same height.[15,24]

Savitzky-Golay Smoothing and Derivation

The most common issue that arise with infrared spectroscopy of biological samples is sloped or oscillatory baselines as a result of scattering and the presence of high frequency instrumentation noise. Typically the first or second derivative is applied to spectra using SG algorithm for the purpose of amplifying spectral variations while minimizing baseline fluctuation. This method can also be used to resolve overlapping bands.[26,27] In classification models, the topographical features of a sample can sometimes be the cause of misclassification but can be minimized through examination of second derivative spectra.[27] To overcome high frequency instrumentation noise, SG smoothing filter reduces the presence of noise by creating an approximate function that attempts to capture important patterns in the data while leaving out noise.[26]

Cropping

Spectral contributions may arise from atmospheric water vapor, carbon dioxide or other interfering compounds. The most common method of reducing the effects of the interfering compounds is to simply ignore the bands that respond to the interfering compounds. This is done by cropping the spectral data to the desired spectral region. The disadvantage of this method is the risk of losing some significant chemical information and producing artifacts at the edges of spectral data.

Mean Centering

Mean centering is typically performed to capture variations in each spectral data to reduce the redundancy in the data. This is performed by subtracting the average spectrum of all spectral data from each individual spectrum. Mean centering data decreases the complexity of the model by reducing the number of factors required to model the data by one.[28]

1.8.2 Partial Least Squares Regression

Partial least squares regression (PLS-R) is a type of multivariate analysis that is aimed at finding a linear relationship between spectral data and classes. PLS-R can be adapted for pattern recognition, giving rise to the partial least squares discriminant analysis (PLS-DA) technique. PLS-DA is a linear two-class classifier that is aimed at finding a straight line, also known as the prediction threshold line, that discriminates between the two specified classes (ie. species in-class versus species out-class). The distance of the samples along the threshold line are known as discriminant score, or prediction of class values. PLS-DA technique is a two step process, as illustrated in Figure 4.

The first step is to train a model to develop a regression model **B** such that the covariance between the training data set ($\mathbf{A_{training}}$) and their subsequent class assignment (**c**) are maximized. The relationship between $\mathbf{A_{training}}$ and **c** can be expressed by

$$\mathbf{c} = \mathbf{A}_{\text{training}}\mathbf{B} \tag{4}$$

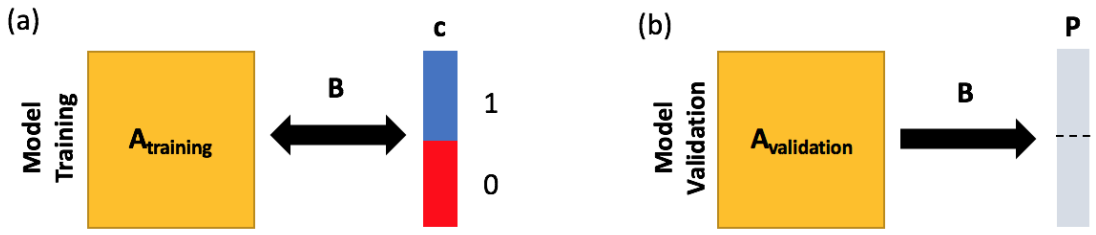

Figure 4. Partial least squares discriminant analysis (PLS-DA) model for two classes. (a) PLS model training using training data set. (b) PLS model validation using validation data set.

where the model **B** is a matrix of loading vectors which contain linear combinations of spectral features that represent all of the variations in the data. Matrix **c** consists of the class assignments for the training set, where 1 is assigned to samples that are in-class and 0 is assigned to samples out-class. Once a trained PLS model is developed, it is used on the validation data set to calculate the discriminant scores (**P**).

$$\mathbf{P} = \mathbf{A}_{\text{validation}}\mathbf{B} \tag{5}$$

Once the discriminant scores are computed, the class a sample belongs to can be determined. In Figure 5 is an illustration of how samples are classified, where the dashed line is the assigned prediction threshold, if the discriminant score is above the threshold line then it would be classified as in-class, if the discriminant score is below the threshold line then it would be classified as out-class. Sample number 6 in Figure 5 is an example of when a sample is incorrectly classified as in-class (false positive).

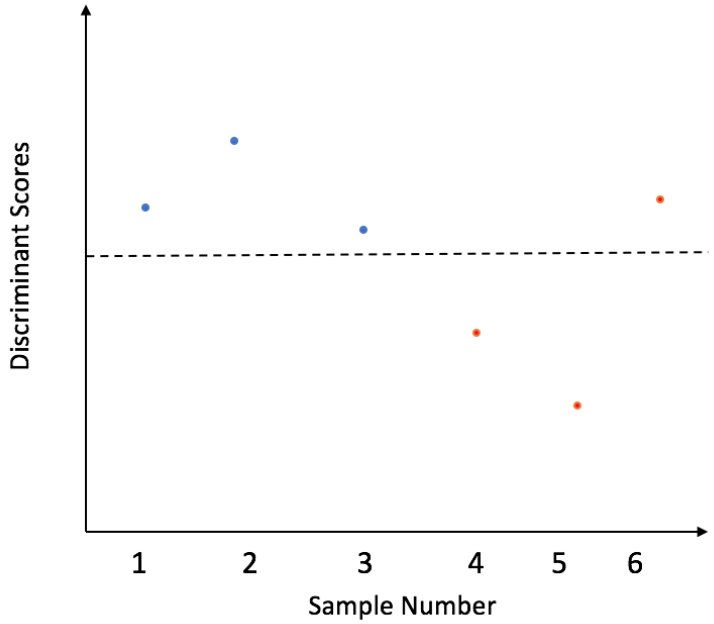

Figure 5. Example of a discriminant score plot with the class prediction threshold line.

1.9 Introduction to Research

Alternative methods to automate mosquito identification and processing are clearly needed to rapidly assess disease vectoring mosquito populations for public health protection. The overall goal is to develop an alternative method that can not only accurately identify mosquito species but also provide other phenotypic characteristics such as virus infection status. The method should be robust, accurate, rapid, economically feasible and easy to use. The prospect of using Fourier transform infrared spectroscopy to discriminate foodborne pathogenic bacteria and detecting *Wolbachia* infection in *Aedes aegypti* mosquitoes motivated us to explore the feasibility of this technique for mosquito identification.

The goal of this research is to show that Fourier transform infrared microspectroscopy, in combination with multivariate modeling, has the required ease of sample preparation, sensitivity and the ability to become a fast and cheap alternative method to current mosquito surveillance

methods. Our spectral acquisition goal is to develop a sampling method that requires little sample preparation and produces reproducible data. For data processing, our goal is to develop a method that optimizes models performance through data pre-processing. This first account of implementation of IR spectroscopy for the speciation of four *Aedes* container inhabiting species: *Ae. aegypti*, *Ae. albopictus*, *Ae. japonicus*, and *Ae. triseriatus*, will hopefully provide operationally useful methods for rapid identification of many more mosquito species.

CHAPTER TWO: EXPERIMENTAL

2.1 Materials

2.1.1 Mosquito Samples

About one hundred forty one adult mosquitoes, male and female, were obtained and preserved in a sealed container at -20 °C. *Ae. japonicas* (46 mosquitoes), *Ae. albopictus* (32 mosquitoes) and *Ae. triseriatus* (49 mosquitoes) were all field collected in North Carolina. *Ae. aegypti* (14 mosquitoes) were lab grown in NOLA. A single mosquito leg is separated from mosquito body and placed on an aluminum sheet, without any pretreatment. A spectrum is collected from the center position of the mosquito's tibia.

2.1.2 Instrumentation

A Nicolet^{TM} model Centaurus infrared microscope was used for all measurements. For each spectrum, parameters used were absorbance in reflection geometry mode, 32 scans (*ca.* 21 sec) were collected with a resolution of 4 cm^{-1}, and spectral range of 4000 to 650 cm^{-1}. A background spectrum is retained for recording 10-15 sample spectra to reduce the effects of constantly changing atmospheric conditions. To reduce the amount of ringing present in the resulting instrumental line shape, Norton-Beer's strong apodizing function was used.[29] The OMNIC^{TM} spectra software was used for IR data acquisition.

2.2 Methods

2.2.1 Measurement Procedure

FT-IR microspectroscopy was used to measure mosquito samples. The general schematic of an FT-IR microscope spectrometer is presented in Figure 2. The IR radiation source passes through the upper objective and projects a conical surface of IR radiation through the sample. Due to the thickness of tibia, light is unable to pass through completely. Therefore, the light must be directed to the surface and returned to the microscope objective by either specular or diffused reflection. After IR radiation is reflected, it follows the same conical surface up through the sample,

back to the objective, and to the detector.

The infrared microscope contains a real time camera which allows for location of the mosquito's tibia on the stage. The microscope is also equipped with knobs that allow for manual adjustment of the stage to focus the infrared light source at an exact position on a mosquito leg sample for better reproducibility. The spectra were acquired at room temperature (20-23 °C). A clean gold-coated microscope slide was used as a reference and 2-3 spectra were recorded across the mosquito tibia to assess the reproducibility of each sample spectrum. However, it was determine that taking just one spectrum at the center of the tibia was sufficient enough to capture chemical information to classify sample.

Once the spectra of each mosquito species were collected, they were saved as a JDX and JCAMP file format. All JDX files were then compiled into a JSON file format which allows for transfer of spectral information, data and meta-data from the instrument computer to a personal computer for ease of subsequent data analysis.

2.2.2 Exploratory Data Analysis

Exploratory analysis is aimed at determining the presence of outliers. It is also a great approach to recognizing patterns in sample distribution such as protein and carbohydrate structure distribution. A band ratio is a simple method for data exploration. Two bands in which absorbance values varied from spectrum to spectrum were selected based on a visual inspection of the infrared spectra between species. The absorbances of these bands were then plotted against each other. The technique as described allows for users to visually inspect data for any patterns in sample distribution of all spectral data.

2.2.3 Data Pre-processing

There are several options for data pre-processing as listed below; however, certain combinations of these steps may be more or less appropriate than others. Thus, to ensure the best data pre-processing procedure was established, a method was developed to calculate the model's performance for several different combinations of data pre-processing steps. The combination of data

pre-processing steps that resulted in the highest accuracy for all mosquito species was used in developing classification model.

Data Pre-processing include:

- A visual inspection of the data to determine spectral regions which capture most of the chemical information in mosquito samples and omit regions that contain redundant or irrelevant chemical information or noise.
- A second derivative Savitzky-Golay algorithm using a window size of 25 cm^{-1} and a second-degree polynomial.
- Data intensity normalized to Amide I.
- Mean centering is performed by subtracting the mean of the calibration spectra from each spectrum.

2.2.4 Model Parameter Optimization

A threshold can be assigned so that the model only assigns a sample to a class if the predicted value is greater than the threshold value. The threshold is optimized to increase the model's classification performance based on calculated accuracy. The number of loading vectors was also optimized to reduce the risk of using more or less than the number of loading vectors needed for classification, which could result in either overfitting or undertraining the model.

To determine the optimum threshold value, we used the code in the appendix, which loops through threshold values between 0.1 and 0.9 (0.05 step size). To optimize the number of loading vectors, an iteration loop steps from 1-25 loading vectors. PLS-DA was performed on all combinations of threshold values and loading vectors, and their subsequent accuracies were calculated.

2.2.5 Model Building and Classification

Once the optimal model parameters (data pre-processing, threshold and number of loading vectors) were defined, the PLS-DA model training set and predict validation set was calculated. For

classification, the library was randomly into two equal sets, a calibration and a validation set. The following steps were performed to determine the classification of each sample:

(i) *PLS regression*: The pre-processed class and spectra of training data set was put into the PLS algorithm and four models were generated, one for each mosquito species.

(ii) *Classification*: The trained model was then applied to the validation data set to obtain discriminant scores for each sample. The sample was assigned to one class when the discriminant score value was above a specific prediction threshold.

2.2.6 Analysis of Model Classification Performance

Once a calibration model was developed, classification parameters such as accuracy, sensitivity, specificity and efficiency were evaluated to asses the models classification performance. When dealing with two classes, a matrix was structured so that the mosquito species in-class were identified as the model's positive, P, value and all other mosquito species were the model's negative, N, value. Four different models were calculated, one for each species. An example of model for *Ae. japonicus* is shown in Table 6.

Figure 6. Confusion matrix of possible model prediction results for *Ae. japonicus*.

		Predicted class	
		Ae. japonicus (P)	All other species (N)
Experimental class	*Ae. japonicus* (P)	TP	FN
	All other species (N)	FP	TN

TP (True Positive) is the number of *Ae. japonicus* samples correctly classified as *Ae. japonicus*, TN (True Negative) is the number of all other samples correctly classified as not *Ae. japonicus*, FN (False Negative) is the number of *Ae. japonicus* samples incorrectly classified as not *Ae. japonicus*, and FP (False Positive) is the number of samples that are not *Ae. japonicus* incorrectly classified as *Ae. japonicus*.

Accuracy was measured by the ratio of correct predictions to the total number of classifications (T): (TP + TN)/T. This describes the proportion of correct classification, independent of the class. A low accuracy indicates that the model is predicting the classifications for each sample incorrectly the majority of the time.[30]

Class sensitivity is the ability for the model to correctly classify mosquito's species to the correct class. This is also known as the measurement of the true positive rate. The class sensitivity was calculated as TP/(TP + FN). Class sensitivity takes values between 0 and 1. If none of the *Ae. japonicus* samples were classified as not *Ae. japonicus* (FN is equal to zero), then the sensitivity for the *Ae. japonicus* class would be equal to 1.[30] A sensitive model is especially important for monitoring newly invasive species, when it is important not to miss identifying them.

In contrast, the class precision measures the capability of the model to correctly identify the mosquito species that don't belong to the modeled class. Precision is calculated as TP/(TP+FP). If none of the samples that are not *Ae. japonicus* were classified as *Ae. japonicus* (FP is equal to zero), the precision would be equal to 1 for the *Ae. japonicus* class.[30] A precise model is important to prevent the "crying wolf" scenario, by reducing the possibility of falsely classifying mosquitoes as a problem species.

CHAPTER THREE: RESULTS AND DISCUSSION

3.1 IR spectra of Mosquito

Using the FT-IR microscope we could measure spectra within seconds that required very little sample preparation and resulted in reproducible spectra using just the mosquito leg. A typical IR spectrum of a mosquito sample is presented in Figure 7. Each mosquito species has a complex biochemical makeup which gives a unique IR spectrum, caused by the stretching and bending vibrations of molecular bonds or functional groups present in its proteins, nucleic acids, lipids, carbohydrates and other smaller molecules. Features in the infrared spectra are affected by the internal and external molecular composition of mosquitoes. Different mosquito species have different biochemical compositions of proteins, carbohydrate, and lipids. Therefore, each mosquito species will have a unique and characteristic chemical fingerprint that is represented in the IR spectrum.

The resulting IR spectra of the mosquito samples tend to be complex and the bands are usually broad due to the superposition of spatial contributions from all the biomolecules present in the mosquito tibia. Thus, deciphering the structures and concentrations of all biomolecules present in a complex biological is difficult and often impossible. Instead, it is the large aggregate of the biomolecular structures that are inspected in order to understand the causes of separation between species. For the purpose of mosquito identification, the following four major absorbance regions are inspected in IR spectra: 3000- 2800 cm^{-1} spectral region is associated with the CH_2 and CH_3 stretching vibration from unsaturated lipids (Figure 7 W1); amide I and amide II bands can be found in the 1700-1500 cm^{-1} region which are attributions of the carbonyl stretching and N-H deformation in proteins (Figure 7 W2); 1500-1200 cm^{-1} is mixed region of hydrocarbon bending vibrations found in lipids, protein, DNA, etc. (Figure 7 W3); C-O-C stretching in the 1190-1000 cm^{-1} region may be the result of vibrations in the chitin structure and/or carbohydrates (Figure 7 W4).[31] Window regions 2 and 4 are the most useful for routine mosquito iden-

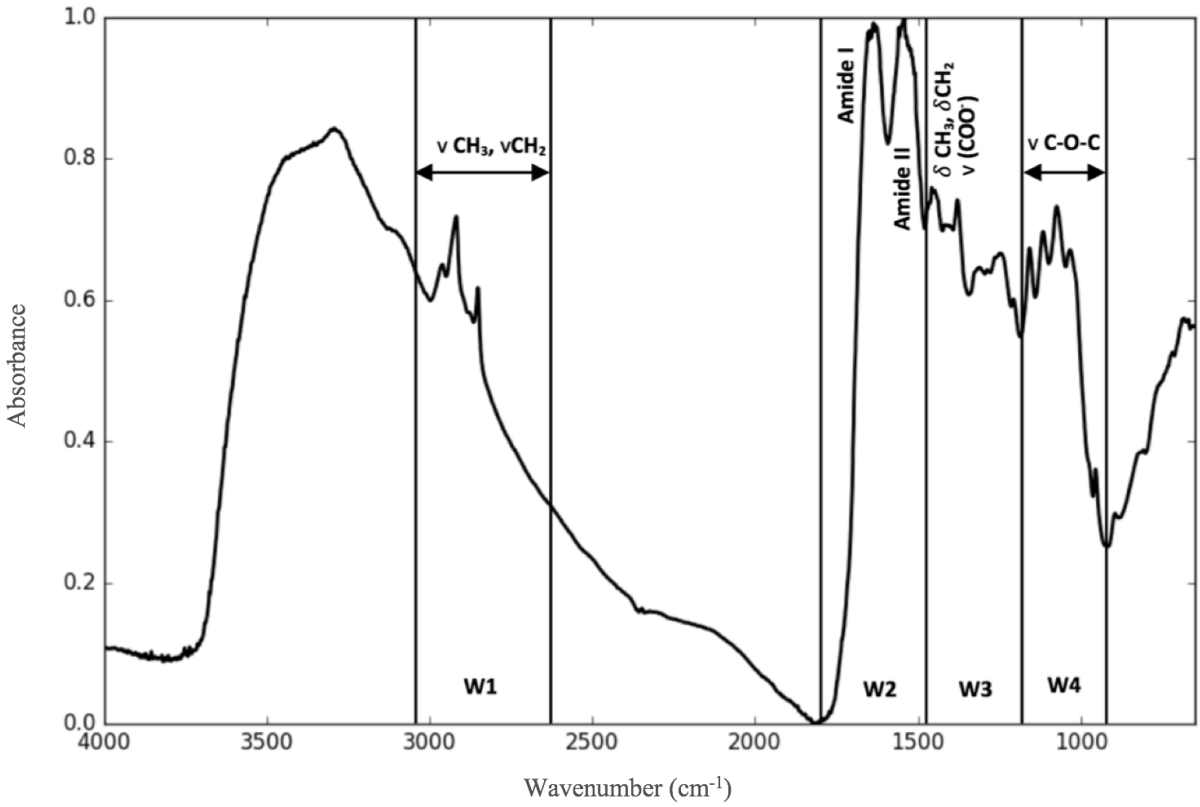

Figure 7. Representative IR spectrum showing peaks from 4,000-650 cm^{-1}, where v = stretching vibration and δ = bending vibration modes. Presented spectrum is of a *Ae. triseriatus* mosquito. W1 - fatty acids, W2 - proteins, W3 - nucleic acids, W4 - carbohydrates.

tification, given that they contain the largest variation in both band intensity and width among mosquito species. Overall the spectra appear very similar for each mosquito species except for some minor changes in the carbohydrate bands ($1190-1000\ cm^{-1}$). Tentative band assignments are given in Table 5.

Although most mosquito spectra look very similar on simple visual examination, subtle quantitative differences such as band intensities and band widths can be observed. Figure 8 shows the average normalized spectra for the four species of mosquitoes studied. Narrower Amide I and Amide II bands present in *Ae. aegypti* spectra are evidence of smaller protein distribution in comparison to other species, while the broadest Amide I and Amide II bands noted in *Ae. albopictus*

Table 1. Assignment of molecular vibrations of functional groups and biomolecular contributor in the mid IR spectra of mosquito.[25,31,32]

Wavenumber (cm^{-1})	Molecular vibrations of functional groups and biomolecular contributor
1670-1622	Amide I band of proteins
1547-1516	Amide II band of proteins
1464	C-H deformation, $>CH_2$ in lipid proteins
1382	C=O symmetric stretching of COO^- group of amino acids, fatty acids
1240	P=O asymmetric stretching in phospholipids
1200 - 900	C-O-C, C-O dominated by ring vibrations in various polysaccharides
1085	P=O symmetric stretching in phospholipids, DNA and RNA

suggest that the *Ae. albopictus* species has a larger variation in its protein structure. Some additional variations in the mean spectra for each species were observed in the C-O stretching region (1200-1000 cm^{-1}) indicative of differences in carbohydrate concentrations. The relation height of the two bands indicated by dashed lines shown in Figure 8(b) vary by different species. For *Ae. aegypti*, the intensity of the band at 1155 cm^{-1} is significantly greater than the band at 1246 cm^{-1}. Similar relative absorbances of the two carbohydrate bands was noted in *Ae. albopictus*. However, the differences in absorbances of the two bands are not as pronounced in comparison to *Ae. aegypti*. For both *Ae. triseriatus* and *Ae. japonicus* the intensity of the bands is reversed for *Ae. Albopictus* and *Ae. aegypti* so that the intensity of 1246 cm^{-1} band is slightly higher than that of 1155 cm^{-1}. Another example of the subtle differences found in the spectra is illustrated in Figure 8(c), where the absorbance ratios of the bands at 1155 cm^{-1} and 1033 cm^{-1} are different for each species.

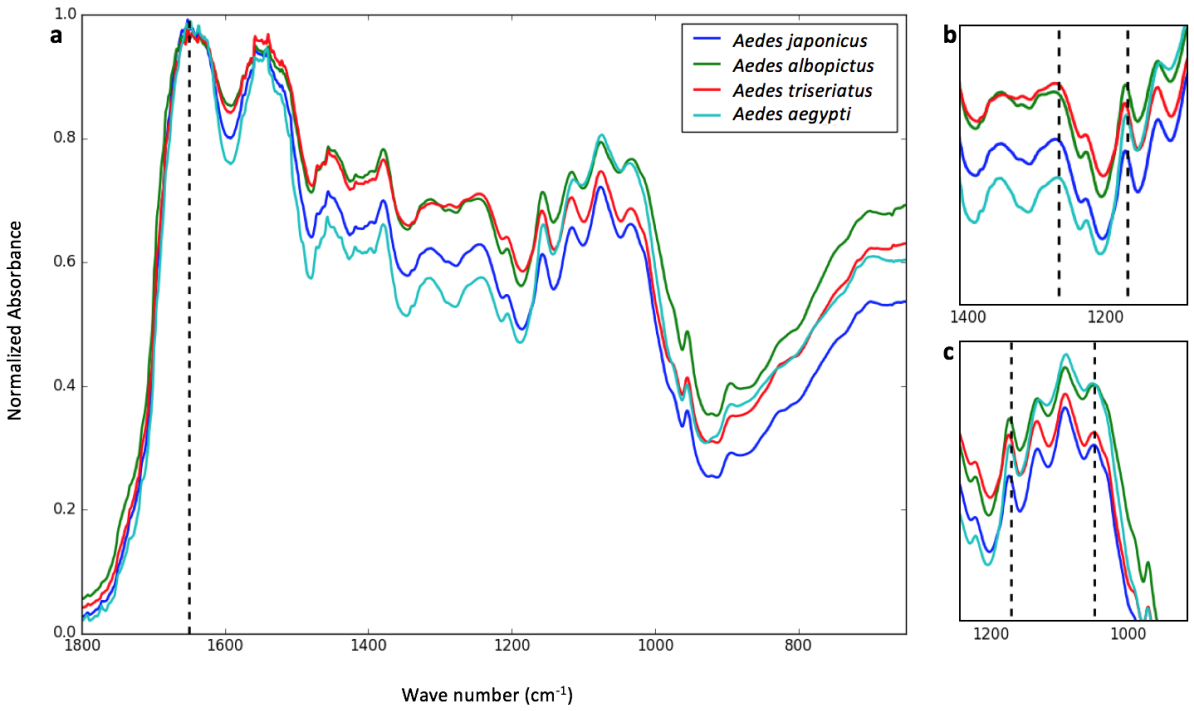

Figure 8. The representative IR spectrum showing peaks from 4,000-650 cm^{-1}. Presented spectrum is of *Ae. aegypti*, *Ae. albopictus*, *Ae. japonicus*, and *Ae. triseriatus* mosquitoes.

The described average spectral variations between species are only a few of the many that may contribute to mosquito identification. To obtain some information on the population distribution of protein and carbohydrate structures occurring within each species, we performed a band ratio of all combinations of the Amide I, the Amide II and four carbohydrate bands of all spectral data.

Figure 9 is a ratio of the absorbance values of amide I and amide II bands. The dashed line before 1600 cm^{-1} is the amide I band and the dashed line after 1600 cm^{-1} is the amide II band. The broad distribution of band ratios values occurring for both *Ae. japonicus* and *Ae. aegypti* suggests a large population distribution of protein structure within the species groups. While a narrower distribution of band ratios occurs for both *Ae. triseriatus* and *Ae. albopictus*, which suggests a smaller population distribution of protein structures. Both *Ae. triseriatus* and *Ae. albopic-*

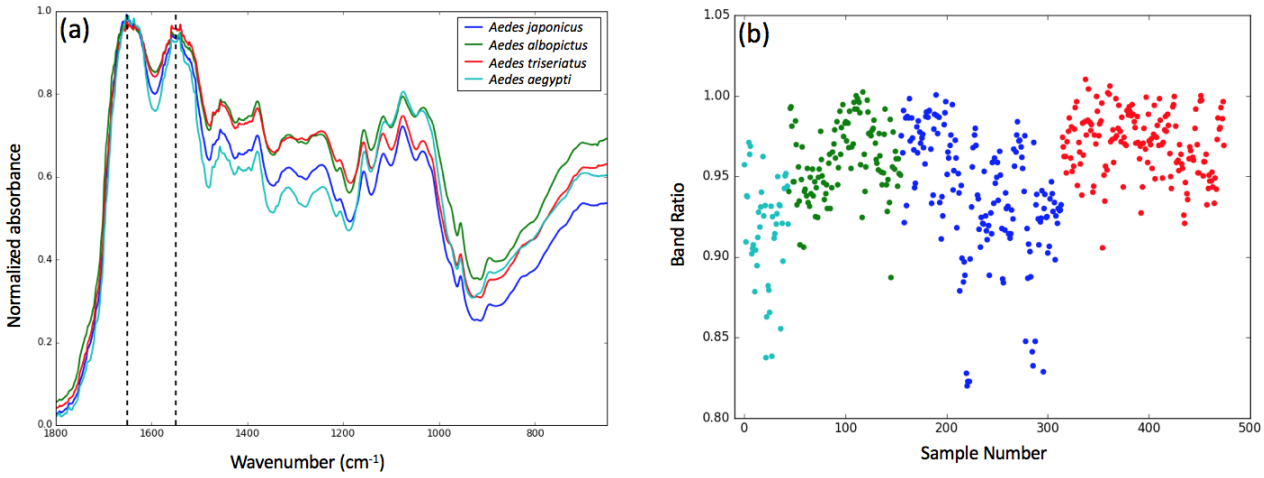

Figure 9. Band ratio of amide I versus amide II bands. (a) Averaged spectra of each mosquito species with dashed lines indicating the location of amide I and amide II bands. (b) Scatter plot of band ratio versus sample number.

tus have similar clustering patterns and band ratios which may also mean that these species could be more closely related in genus than the other groups. Some samples from the *Ae. japonicus* group are oddly separated from the rest of the samples, indicating either outliers or misclassified cryptic species.

Figure 10 is the amide I band and all the carbohydrate bands indicated by the labels a, b, c and d. Each scatter plot is the ratio of the absorbance of the Amide I band against each carbohydrate band absorbance. With this simple band ratio chemical variations among mosquito species can be observed. Primarily, the distribution of band ratio values within each group varies greatly among species. However, the results when compared to the ratio of amide I with a different carbohydrate band do not vary very much. Other scatter plots were developed to compare the ratios of the Amide II band against all carbohydrate bands, and ratios of all carbohydrate bands against each other. Similar variations in the distribution of the band ratios were observed in each scatter plot. Although we can see some variations that are occurring among species from a simple band ratio, comparison of just two bands will not be enough to capture all variations occurring in

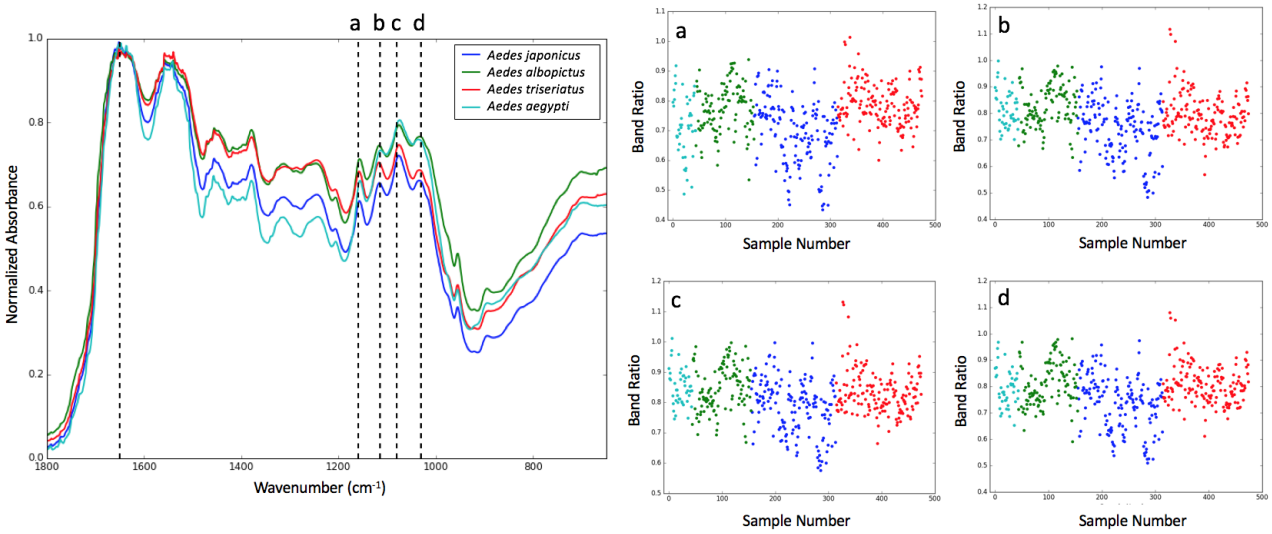

Figure 10. Left is the averaged spectra of each mosquito species with dashed lines indicating the location of Amide I band and each of the four carbohydrate bands. Right is the band absorbance ratios for Amide I and four carbohydrate bands.

complex spectra of a mosquito sample. Therefore, for the purpose of identification, multivariate analysis was employed.

3.2 Mosquito Data Pre-processing Results and Optimization

To optimize the best data pre-processing (PP) steps, we preprocessed our datasets using different orderings between the following four steps: normalization, Savitzky-Golay differentiation, mean centering and cropping. Partial least square discriminant analysis was performed on each preprocessed test dataset, and their subsequent accuracies were calculated. The Table 3 contains all combinations of data pre-processing that were explored and the model's performance based on calculated accuracy.

In the table, **A** is the spectral data, **c** is cropping, **m** is mean centering, **d** is the second derivative, and **n** is normalization. The cells highlighted in yellow represent the data pre-processing steps which performed the best for that species. The orange highlighted cells are the data pre-processing steps that performed the worst for each species. Part of optimizing the data pre-processing

Table 2. Combinations of data pre-processing steps explored for each species and the models performance based on calculated accuracies.

#	PP	*Ae. japonicus*	*Ae. albopictus*	*Ae. triseriatus*	*Ae. aegypti*
1	Acn	79.8	95.8	90.3	99.6
2	Amc	81.5	93.7	87.4	96.6
3	A	81.5	93.7	87.4	96.6
4	Acnm	79.8	95.8	90.3	99.6
5	Acndm	93.3	99.6	97.5	100
6	Amcd	89.9	99.2	96.2	100
7	Amcn	81.1	92.9	86.6	98.3
8	Acnd	93.3	99.6	97.5	100
9	Am	81.5	93.7	87.4	96.6
10	Acd	89.9	99.2	96.2	100
11	Ac	81.5	93.7	87.4	96.6
12	Amcdn	91.6	96.6	92.9	100
13	Acnmd	93.3	99.6	97.5	100
14	Acdn	93.3	98.7	96.6	100

steps was to determine whether the order in which the data pre-processing steps are performed caused any significant impact on the performance of the model. A comparison of methods 5 and 13, where the second derivative was performed before mean centering in method 5 and in method 13 the second derivative is performed after mean centering shows that although the order of pre-processing steps is different, the calculated accuracies are the same for both . However, for methods 8 and 14, where normalization is performed before derivation in method 8 and vice versa for method 14, the accuracies are slightly lower for *Ae. albopictus* and *Ae. triseriatus* models when the second derivative is performed before normalization. Upon inspection, it is evident that the only ordering of data pre-processing that impacts the model's calculated accuracy is the order in which normalization and the second derivative are performed. Normalization should always be performed prior to performing a second derivative to achieve the most accurate results. The result of the optimization of data pre-processing concludes that the optimal model is achieved when the spectra is first cropped between regions 1800 - 400 cm^{-1}, then normalized to Amide I band, followed by a Savitzky-Golay second derivative and mean centered, as described in method 13.

3.3 Threshold Optimization

As described previously, PLS-DA models classify samples on the basis of discriminant score values. A threshold can be assigned so that the model only assigns a sample to a class if the discriminant score value is greater than the threshold value. Therefore, it is important to ensure the threshold is optimized to achieve the highest accuracy. To accomplish this goal, an iteration loop chooses threshold values between 0.3 to 0.9 (0.05 step size), all other parameters held constant, subsequent accuracies were calculated. Their accuracy as a function of threshold is shown in Figure 11. The optimal prediction threshold value is chosen by the one that resulted in the highest accuracy for each class. The dashed line in Figure 11 indicates the prediction threshold value that resulted in the maximum accuracy for all groups is 0.5.

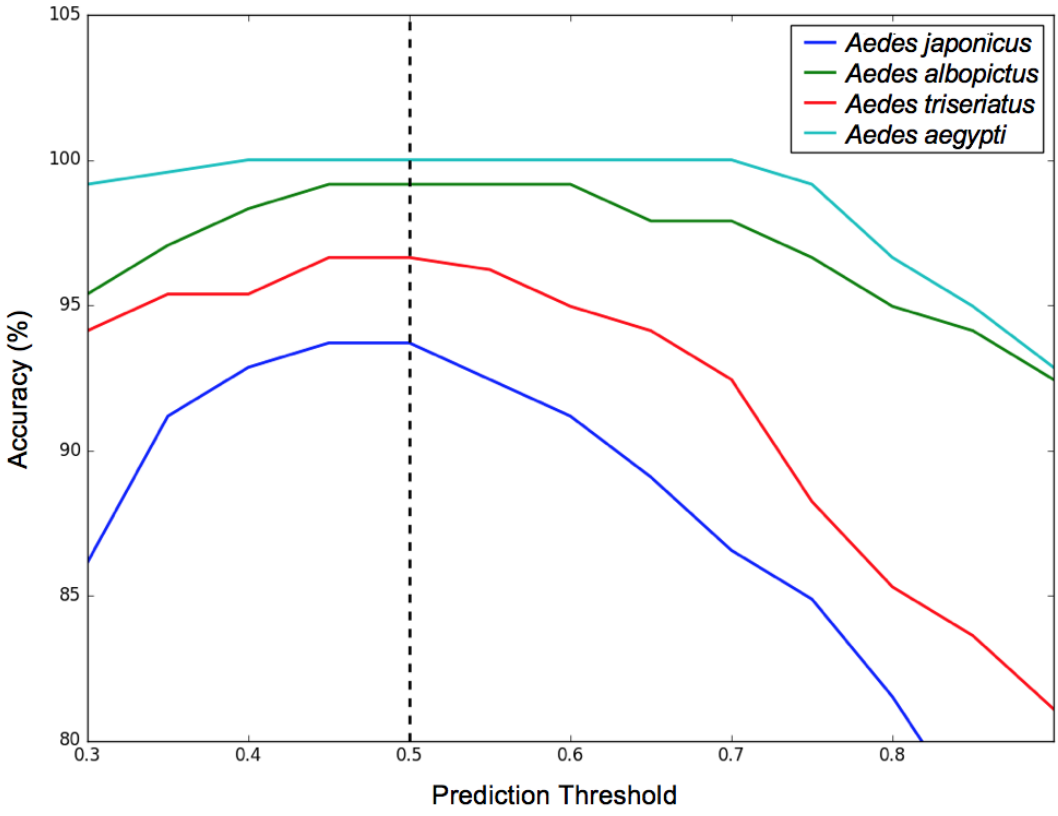

Figure 11. Models performance based on calculated accuracy at different threshold values for each of the four mosquito species.

3.4 Loading Vector Optimization

The number of loading vectors (nlv) in the PLS algorithm which allowed the fitting of the complex system without overfitting or undertraining was optimized. An iteration loop chooses from a number of loading vectors between 1 to 25 with a step size of 1. PLS-DA was performed on each number of loading vectors with all other parameters held constant and the subsequent accuracy was calculated. The optimal number of loading vectors is chosen by the number that resulted in the highest accuracy for each class. Table 3 shows the results.

Table 3. Different number of loading vectors used in PLS-DA model for each mosquito species and the models performance based on calculated accuracies.

nlv	*Ae. japonicus*	*Ae. albopictus*	*Ae. triseriatus*	*Ae. aegypti*
1	72.3	79.4	68.1	92.8
2	74.4	85.3	71.8	91.6
3	77.7	92.4	83.2	97.9
4	79.4	90.7	86.5	98.7
5	83.6	97.0	91.6	99.6
6	87.8	97.5	92.4	100
7	89.1	98.3	93.7	100
8	88.2	98.7	96.6	100
9	89.9	99.2	96.2	100
10	92.4	99.2	95.8	100
11	92.9	99.2	95.8	100
12	90.3	99.6	95.4	100
13	92.0	99.2	96.2	100
14	92.4	99.2	95.8	100
15	91.2	98.7	94.9	100
16	91.6	98.3	94.5	100
17	91.2	98.7	95.4	99.2
18	91.6	98.3	94.9	99.2
19	92.0	97.9	94.1	99.6
20	89.9	97.9	93.3	99.6
21	90.3	97.9	92.9	99.6
22	89.5	97.5	92.9	99.2
23	89.1	97.5	93.3	99.2
24	88.6	97.1	92.9	99.6

The cells highlighted in yellow indicate the number of loading vectors which yielded the highest accuracy for that group. The orange highlighted cells are the number of loading vectors which yielded the lowest accuracy for that group. Using less that 10 loading vectors does not capture all of the variation required to minimize the number of mosquito samples classified incorrectly, resulting in an undertrained model. However, using more than 10 loading vectors the accuracy slightly increases for *Ae. japonicus* before it starts decreasing for all classes. Thus, to prevent overfitting the model, 10 loading vectors were used.

3.5 Mosquito Classification by the Partial Least Squares Discriminant Analysis (PLS-DA) Classification Model

The optimized partial least squares discriminant analysis was applied to the training set, the developed models were then validated using validation set samples. The training and validation sets were preprocessed using data pre-processing method 13 and 10 loading vectors were chosen to build the classification model. The decision rule for classification is defined by the assigned threshold, which was also optimized and set to 0.5 for each class.

The class predictions calculated by the PLS-DA model for the validation sets are shown in Figure 12. The PLS regression model predicts the value of the validation samples, then a threshold call is made where prediction value above the threshold are assigned to the species and the samples below the threshold are not in the species. The threshold is indicated by the dashed horizontal line in Figure 12. Field collected mosquitoes (*Ae. japonicus* (a), *Ae. triseriatus* (b), *Ae. albopictus* (c)) have greater distribution in prediction values than laboratory reared *Ae. aegypti* (d) mosquitoes. This could suggest the presence of contamination from environmental debris or chemical alteration of the tibia. However, the models are still able to classify mosquitoes to their correct class with high accuracy, demonstrating the robustness of the developed method. In *Ae. triseriatus* model (c), a false positive appears much higher than all the other predicted samples. This may actually be an accurate prediction for that sample and the misclassification may be due to mistaken morphological determination. A summary of performance is given in Table 4.

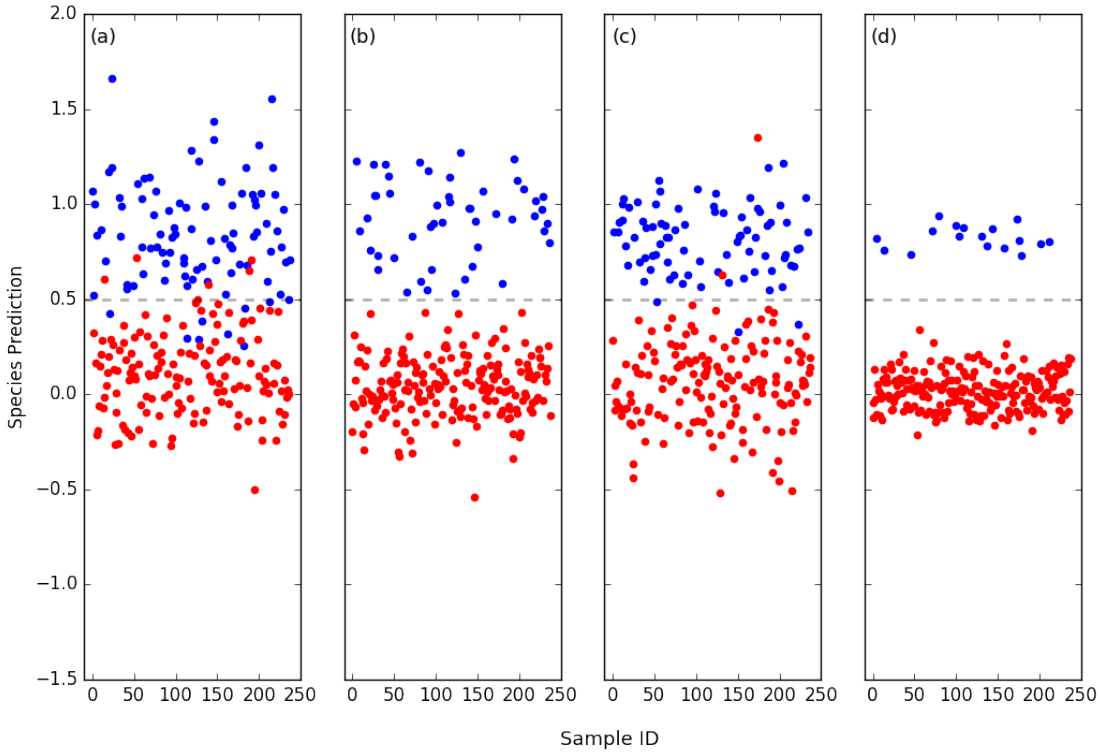

Figure 12. Results of training set of the PLS-DA model for (a) *Ae. japonicus*, (b) *Ae. albopictus*, (c) *Ae. triseriatus*, and (d) *Ae. aegypti*.

Table 4. A summary of PLS-DA classification performance.

	Ae. japonicus	*Ae. triseriatus*	*Ae. albopictus*	*Ae. aegypti*
Total Tested	92	49	80	17
True Positive	84	77	49	17
False Positive	5	2	0	0
True Negative	141	156	189	221
False Negative	8	3	0	0

Most misclassifications occurs in the *Ae. japonicus* model, where five samples (red) were incorrectly classified as being *Ae. japonicus* and eight *Ae. japonicus* samples (blue) were classified incorrectly as not being *Ae. japonicus*. Since this species of mosquitoes were the first to be stud-

ied, the misclassifications may be caused by faulty spectra, resulting from the excessive presence of water vapor or chemical contamination in the lab while handling the mosquitoes prior to measurement. However, other causes may have resulted in misclassification such as environmental contaminations, mistaken morphological determination by entomologist or the presence of an unknown cryptic species. Thus, to improve the classification performance of the model, inspection of the misclassified spectral data should be performed to determine the cause of misclassification.

3.6 Classification Performance

The performance of the classification models were evaluated by calculated accuracy, sensitivity and precision from the results in Table 4. The results are summarized in Table 5. The validation sets were predicted with the accuracies for all classes equal to or greater than 95%. This indicates that the method developed performs very well to predict mosquito identification correctly, independent of the class. The PLS models for each class gave sensitivity values between 91% to 100%. This indicates that the model can predict the presence of certain mosquito species every time they are present. This is especially important when when monitoring for the presence of public health important invasive species such as *Ae. aegypti* which is responsible for the transmission of Zika.[5,6] The model's precision was also calculated and was between 94.4% to 100% for all classes, indicating that the model can be trusted to give true predictions of the presence of certain species, reducing the chances of having false positive predictions (crying wolf scenario).

Table 5. Classification performance criteria obtained for PLS-DA models of each mosquito species.

Calculations	*Ae. japonicus*	*Ae. triseriatus*	*Ae. albopictus*	*Ae. aegypti*
Accuracy (%) = (TP+TN)/T	94.5	97.9	100	100
Sensitivity (%) = TP/(TP+FN)	91.3	96.2	100	100
Precision (%) = TP/(TP+FP)	94.4	97.5	100	100

CHAPTER FOUR: CONCLUSIONS AND FUTURE DIRECTIONS

This study has demonstrated that mid-infrared spectroscopy, coupled with partial least square discriminant analysis, can be used as a technique for discrimination among different mosquito species. The mid-infrared spectrum contains information useful for discriminating among mosquito species based on variations in biochemical compositions, when combined with PLS-DA models. In summary, (i) leg samples of mosquitoes were successfully used to speciate mosquitoes with very little sample preparation required. (ii) Data pre-processing is recommended and was found to improve classification performance when both normalization and second derivation using the Savitzky-Golay algorithm was performed. (iii) Optimizing the number of loading vectors used for classification also improved the model's performance by reducing the risk of overfitting or undertraining.

The extremely high sensitivity and precision of the developed models show the potential of this technique as a rapid and easy method for speciation of both laboratory and field caught mosquitoes. Although not confirmed, it appears that the main contribution to discrimination is the variation in both the protein and carbohydrate regions of the IR spectrum. The FT-IR microspectroscopy can achieve an accuracy of 94%-100% for differentiation of four *Aedes* mosquitoes: *Ae. aegypti* 100% (N = 17), *Ae. albopictus* 100% (N = 49), *Ae. japonicus* 94.5 % (N = 92), and *Ae. triseriatus* 97.9% (N = 80), compared to a trained entomologist. Although there are larger inherent variations in field caught mosquitoes compared to lab reared mosquitoes, the models is still able to classify all mosquito species with high accuracy, demonstrating the robustness of the technique.

Improved quality control, data pre-processing, model parameters, and the assessment of misclassified samples will result in the development of more robust and accurate models. Larger data sets and the application of different algorithms such as support vector machines (SVM) and artificial neural networks (ANN) will likely improve the accuracy and further improve the sensitivity

and specificity of the technique. The process of sample acquisition and spectral recording takes less than 1 minute per sample using a single IR microscope, enabling approximately 5,000 samples to be processed in a week (12 h day) with a single instrument. Our results show real potential that a viable automatic spectroscopic alternative to mosquito surveillance can be developed, with efficiencies made in time, cost, and training of personnel.

REFERENCES

[1] Vector-borne Diseases Fact sheet. 2017; http://www.who.int/mediacentre/factsheets/fs387/en/.

[2] Barker, C.; Collins, C.; Colon, J.; Rutledge Connelly, C.; Debboun, M.; Dormuth, E.; Faraji, A.; Fujioka, K.; R. Lesser, C.; Michaels, S.; Schankel, B.; Smith, K.; Unlu, I.; Bruce White, G. *Best Practices for Integrated Mosquito Management: A Focused Update*; 2017; pp 1–58.

[3] Davies, M.; Foust, E.; Williams, C.; Watkins, H.; Michael, L.; Zimmerman, S. North Carolina Vector Borne Disease Management. 2016; http://epi.publichealth.nc.gov/cd/vector/VectorborneDiseaseProgramWhitePaper.pdf.

[4] Eldridge, B. F. Strategies for surveillance, prevention, and control of arbovirus diseases in western North America. *American Journal of Tropical Medicine and Hygiene* **1987**, *37*, 77–86.

[5] Grard, G.; Caron, M.; Mombo, I. M.; Nkoghe, D.; Mboui Ondo, S.; Jiolle, D.; Fontenille, D.; Paupy, C.; Leroy, E. M. Zika Virus in Gabon (Central Africa) 2007: A New Threat from *Aedes albopictus*? *PLOS Neglected Tropical Diseases* **2014**, *8*, 1–6.

[6] Li, C. X.; Guo, X. X.; Deng, Y. Q.; Xing, D.; Sun, A. J.; Liu, Q. M.; Wu, Q.; Dong, Y. D.; Zhang, Y. M.; Zhang, H. D.; Cao, W. C.; Qin, C. F.; Zhao, T. Y. Vector competence and transovarial transmission of two *Aedes aegypti* strains to Zika virus. *Emerging Microbes and Infections* **2017**, *6*, 1–7.

[7] Bhatt, S.; Gething, P.; Brady, O.; Messina, J.; Farlow, A.; Moyes, C. The global distribution and burden of dengue. *Nature* **2012**, *496*, 504–507.

[8] Hernandez-Triana, L. M.; Jeffries, C. L.; Mansfield, K. L.; Carnell, G.; Fooks, A. R.; Johnson, N. Emergence of West Nile Virus Lineage 2 in Europe: A Review on the Introduction and Spread of a Mosquito-Borne Disease. *Frontiers in Public Health* **2014**, *2*, 1–8.

[9] Bewick, S.; Agusto, F.; Calabrese, J. M.; Muturi, E. J.; Fagan, W. F. Epidemiology of La Crosse Virus Emergence, Appalachia Region, United States. *Emerging Infectious Diseases* **2016**, *22*, 1921–1929.

[10] Thiboutot, M. M.; Kannan, S.; Kawalekar, O. U.; Shedlock, D. J.; Khan, A. S.; Sarangan, G.; Srikanth, P.; Weiner, D. B.; Muthumani, K. Chikungunya: A potentially emerging epidemic? *PLoS Neglected Tropical Diseases* **2010**, *4*, 1–8.

[11] Sikulu, M.; Killeen, G. F.; Hugo, L. E.; Ryan, P. A.; Dowell, K. M.; Wirtz, R. A.; Moore, S. J.; Dowell, F. E. Near-infrared spectroscopy as a complementary age grading and species identification tool for African malaria vectors. *Parasites and Vectors* **2010**, *3*, 1–7.

[12] Müller, P.; Pflüger, V.; Wittwer, M.; Ziegler, D.; Chandre, F.; Simard, F.; Lengeler, C. Identification of Cryptic *Anopheles* Mosquito Species by Molecular Protein Profiling. *PLoS ONE* **2013**, *8*.

[13] Krajacich, B. J.; Meyers, J. I.; Alout, H.; Dabire, R. K.; Dowell, F. E.; Foy, B. D. Analysis of near infrared spectra for age-grading of wild populations of *Anopheles gambiae*. *Parasites & Vectors* **2017**, *10*, 552–565.

[14] Agelet, L. E.; Hurburgh, C. R. A tutorial on near infrared spectroscopy and its calibration. *Critical Reviews in Analytical Chemistry* **2010**, *40*, 246–260.

[15] Baker, M. J. et al. Using Fourier transform IR spectroscopy to analyze biological materials. *Nature Protocols* **2014**, *9*, 1771–1791.

[16] Stuart, B. H. *Infrared Spectroscopy: Fundamentals and Applications*; John Wiley & Sons, 2005; Vol. 1; pp 25–33, 35.

[17] Kazarian, S.; Chan, K. Applications of ATR-FTIR spectroscopic imaging to biomedical samples. *Biochimica et Biophysica Acta - Biomembranes* **2006**, *1758*, 858 – 867.

[18] Mitchell, M. B. *Structure-Property Relations in Polymers*; American Chemical Society, 1993; Vol. 236; pp 351–375.

[19] Khoshmanesh, A.; Christensen, D.; Perez-Guaita, D.; Iturbe-Ormaetxe, I.; O'Neill, S. L.; McNaughton, D.; Wood, B. R. Screening of *Wolbachia* endosymbiont infection in *Aedes aegypti* mosquitoes using attenuated total reflection mid-Infrared spectroscopy. *Analytical Chemistry* **2017**, *89*, 5285–5293.

[20] Theophilou, G.; Lima, K. M. G.; Martin-Hirsch, P. L.; Stringfellow, H. F.; Martin, F. L. ATR-FTIR spectroscopy coupled with chemometric analysis discriminates normal, borderline and malignant ovarian tissue: classifying subtypes of human cancer. *The Analyst* **2016**, *141*, 585–594.

[21] Davis, R.; Mauer, L. Fourier transform infrared (FT-IR) spectroscopy: A rapid tool for detection and analysis of foodborne pathogenic bacteria. *Current Research, Technology and Education Topics in Applied Microbiology and Microbial Biotechnology.* **2010**, *2*, 1582–1594.

[22] Zhang, Y. P.; Lewis, R.; McElhaney, R. N.; Hodges, R. S. Interaction of a peptide model of a hydrophobic transmembrane α-helical segment of a membrane protein with phosphatidylcholine bilayers: Differential scanning calorimetric and FTIR spectroscopic studies. *Biochemistry* **1992**, *31*, 11579–11588.

[23] Cozzolino, D.; Holdstock, M.; Dambergs, R. G.; Cynkar, W. U.; Smith, P. A. Mid infrared

spectroscopy and multivariate analysis: A tool to discriminate between organic and non-organic wines grown in Australia. *Food Chemistry* **2009**, *116*, 761–765.

[24] Zarnowiec, P.; Lechowicz, L.; Czerwonka, G.; Kaca, W. Fourier transform infrared spectroscopy (FTIR) as a tool for the identification and differentiation of pathogenic bacteria. *Current Medicinal Chemistry* **2015**, *22*, 1710–1718.

[25] Movasaghi, Z.; Rehman, S.; Rehman, I. U. Fourier transform infrared (FTIR) spectroscopy of biological tissues. *Applied Spectroscopy Reviews* **2008**, *43*, 134–179.

[26] Vivó-Truyols, G.; Schoenmakers, P. J. Automatic selection of optimal Savitzky-Golay smoothing. *Analytical Chemistry* **2006**, *78*, 4598–4608.

[27] Cao, J.; Ng, E. S.; McNaughton, D.; Stanley, E. G.; Elefanty, A. G.; Tobin, M. J.; Heraud, P. Using Fourier transform IR spectroscopy to analyze biological materials. *Journal of Biophotonics* **2014**, *7*, 767–781.

[28] Haaland, D. M.; Thomas, E. V. Partial least-squares methods for spectral analyses. 1. Relation to other quantitative calibration methods and the extraction of qualitative information. *Analytical Chemistry* **1988**, *60*, 1193–1202.

[29] Tahic, M. K.; Naylor, D. a. Apodization Functions for Fourier Transform Spectroscopy. *Optical Society of America* **2007**, *24*, 3644–3648.

[30] Da Silva Campos, N.; De Sá Oliveira, K.; Almeida, M. R.; Stephani, R.; De Oliveira, L. F. C. Classification of frankfurters by FT-Raman spectroscopy and chemometric methods. *Molecules* **2014**, *19*, 18980–18992.

[31] Coates, J. Interpretation of infrared spectra, a practical approach. *Encyclopedia of Analytical Chemistry* **2006**,

[32] Socrates, G. *Infrared and Raman characteristic group frequencies*; Wiley, 2004; Vol. 35; p 366.

APPENDIX

This is a code for performing the following data analysis: * Preprocessing * Optimization * Band Ratio * Partial Least Sqaures DA * Plotting and inspecting spectra and average spectra

Setup compute environment

```
In [1]: import numpy as np
        from numpy.linalg import pinv,inv
        import matplotlib as mpl
        ##this makes it rotatable when not in jupyter
        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt
        %matplotlib inline
        %matplotlib notebook
        import pandas as pd
        import math
        from pandas.tools.plotting import scatter_matrix
        from pandas.tools.plotting import table


        import sys
        sys.path.append('./lib/')
        import analtool3 as AT
        from chemo import svdpcaf,fovcalc

        fighole = '../skeetertype/Thesis/figures/'
        datahole = './datasets/'
        lw = 2
        lsz = 12
        outputdpi = 100
        mpl.rcParams['xtick.labelsize'] = lsz
        mpl.rcParams['ytick.labelsize'] = lsz
        mpl.rcParams['font.size'] = lsz
        mpl.rcParams['lines.linewidth'] = lw
```

Local definitions: graphics and preprocessing functions

```
In [2]: def getcolors():
            '''
            This function exports color names that are
            usable in python.
            Use this to generate consistant
            colors for graphs and what not.
            Usage:  colornames = getcolors()
            colornames is a list of color named strings
            '''
            colornames = ['b', 'g', 'r', 'c', 'm', 'y',
```

```
  'k', 'w']

    return colornames

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    import itertools
    fig, ax = plt.subplots(figsize=(9,9),nrows=1, ncols=1,
    sharex=False)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45,
    horizontalalignment="right")
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(
    range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh
                 else "black")

    plt.tight_layout()
    plt.ylabel('Morphologically Determined Species')
    plt.xlabel('Predicted Species')
    return fig

def meancenter(A):
    '''
    This function mean centers every spectrm in an array.
```

```
    Usage: Amc = meancenter(A)
    A and Amc are both arrays (number of spectra, number
    of wavenumbers).
    the mean is determined with in the function.
    '''
    Amean = A.mean(axis=0)
    nspec,npts = A.shape
    Anew = np.zeros((nspec,npts),dtype=float)
    for i in range(nspec):
        Anew[i,:] = A[i,:] - Amean
    return Anew

def crop(A,hi,lo):
    '''
    This function crops the data to high and low limits.
  Note that hi and lo are indices not wavenumber or wavelengths.
    Usage: Ac = crop(A, hi,lo)
    both A and Ac are arrays whose dems. are (nspec,nwavelengths)
    '''
    crng = np.arange(lo,hi)
    Ac = A[:,crng]
    return Ac

def getgrpmean(A,truearray):
    '''
    This function generates a group mean spectra.
    Usage:  Agrpmean = getgrpmean(A,grps)
    A and Agrpmean are both arrays.
    Agrpmean(ngrps,nwavelengths) and A(nspec,nwavelengths)
    grps is a vector (nspec,) that contatains group integer
    labels each spectrum in A.
    '''
    nspec,npts = A.shape
    ngrps = len(np.unique(truearray))
    Amean = np.zeros((ngrps,npts),dtype=float)
    for i in np.unique(truearray):
        Amean[i,:] = A[truearray == i,:].mean(axis=0)
    return Amean

#Second Derivative
def getgrpsd(A,truearray):
    '''
    This function generates a group mean spectra.
    Usage:  Agrpmean = getgrpmean(A,grps)
    A and Agrpmean are both arrays.
    Agrpmean(ngrps,nwavelengths) and A(nspec,nwavelengths)
    grps is a vector (nspec,) that contatains group integer
    labels each spectrum in A.
```

```
    '''
    nspec,npts = A.shape
    ngrps = len(np.unique(truearray))
    Amean = np.zeros((ngrps,npts),dtype=float)
    for i in np.unique(truearray):
        Asd[i,:] = AT.sg(A[truearray == i,:],25,2,deriv=2,
        rate=1)
    return Asd

def getgrpmax(A,truearray):
    '''
    This function generates a group mean spectra.
    Usage:  Agrpmean = getgrpmean(A,grps)
    A and Agrpmean are both arrays.
    Agrpmean(ngrps,nwavelengths) and A(nspec,nwavelengths)
    grps is a vector (nspec,) that contatains group integer
    labels each spectrum in A.
    '''
    nspec,npts = A.shape
    ngrps = len(np.unique(truearray))
    Amax = np.zeros((ngrps,npts),dtype=float)
    for i in np.unique(truearray):
        Amax[i,:] = A[truearray == i,:].max(axis=0)
    return Amax

def getgrpmin(A,truearray):
    '''
    This function generates a group mean spectra.
    Usage:  Agrpmean = getgrpmean(A,grps)
    A and Agrpmean are both arrays.
    Agrpmean(ngrps,nwavelengths) and A(nspec,nwavelengths)
    grps is a vector (nspec,) that contatains group integer
    labels each spectrum in A.
    '''
    nspec,npts = A.shape
    ngrps = len(np.unique(truearray))
    Amin = np.zeros((ngrps,npts),dtype=float)
    for i in np.unique(truearray):
        Amin[i,:] = A[truearray == i,:].min(axis=0)
    return Amin

def getgrpstd(A,truearray):
    '''
    This function generates a group mean spectra.
    Usage:  Agrpmean = getgrpmean(A,grps)
    A and Agrpmean are both arrays.
    Agrpmean(ngrps,nwavelengths) and A(nspec,nwavelengths)
    grps is a vector (nspec,) that contatains group integer
```

```
        labels each spectrum in A.
        '''
        nspec,npts = A.shape
        ngrps = len(np.unique(truearray))
        Astd = np.zeros((ngrps,npts),dtype=float)
        for i in np.unique(truearray):
            Astd[i,:] = A[truearray == i,:].std(axis=0)
        return Astd
```

Model evaluation functions

```
conf90=1.64
conf95=1.96
conf98=2.33
conf99=2.58
def calc_qual(T,tp,tn,fp,fn):
    '''
    calculates model performance parameters (selectivity,
    sensitivity, mathews coeff., precision, accuracy and
    efficiency) from confusion matrix.
    '''
    if len(T) == tp+tn+fp+fn:
        pass
    else:
        print('WARNING:  Integrety check:  ',len(T),' != ',
        tp+tn+fp+fn,'FAIL')

    error = (fp+fn)/len(T)
    h = conf95*math.sqrt((error*(1-error))/(nspec/2)) *100

    sensitivity = tp/(tp+fn) *100
    precision = tp/(tp+fp) *100
    accuracy = (tp+tn)/len(T) *100
    #efficiency = ((sensitivity+specificity)/2)
    mc = ((tp*tn)-(fp*fn))/np.sqrt((tp+fp)*(tp+fn)*(tn+fp)*
    (tn+fn))
    repdict = {'sensitivity':sensitivity,'precision':precision,
               'accuracy':accuracy,'mc':mc, '+/-': h}
    df = pd.DataFrame.from_dict(repdict,orient='index')
    return df
```

Load data

```
In [9]: dsname = './datasets/dataset_20170710.json'
        mdsname = './datasets/mdataset_20170710.json'
        ort = 'split'
        df = pd.read_json(dsname,typ='frame',orient=ort)
        df.columns.name = 'file'
```

```
mdf= pd.read_json(mdsname,typ='frame',orient=ort)
mdf.columns.name = 'file'

##select on leg
leg = 3
df = df[mdf['leg'] == leg]
mdf = mdf[mdf['leg'] == leg]

##select on sex
## turn this off or on
sexflag = 'on'
if sexflag == 'on':
    sex = 'f'
    df = df[mdf['sex'] == sex]
    mdf = mdf[mdf['sex']==sex]

specidx = df.index

## Data Preprocessing ##

xorg = np.array(df.columns.tolist())

nspec,npts = df.shape
A = df.as_matrix()


#Meancentering
Amc =  meancenter(df.as_matrix())

### crop data, remember these are indices not wavelengths
hi = 3350
lo = 0
Ac = crop(A,hi,lo)
Amcc = crop(Amc,hi,lo)
x = xorg[lo:hi]


#### derivative spectra of cropped data
nspec,npts = Ac.shape
Acd = np.zeros((nspec,npts),dtype=float)
Amccd = np.zeros((nspec,npts),dtype=float)
for i in range(nspec):
    Acd[i,:] = AT.sg(Ac[i,:],25,2,deriv=2,rate=1)
    Amccd[i,:] = AT.sg(Amcc[i,:],25,2,deriv=2,rate=1)

#### normalize all cropped and derivatized data
normtype = 1
nspec,npts = Ac.shape
```

```
Acn = np.zeros((nspec,npts),dtype=float)
Acnmc = np.zeros((nspec,npts),dtype=float)
Amccn = np.zeros((nspec,npts),dtype=float)
Acdn = np.zeros((nspec,npts),dtype=float)
Amccdn = np.zeros((nspec,npts),dtype=float)
for i in range(nspec):
    Acn[i,:] = AT.hnorm(Ac[i,:],normtype)
    Amccn[i,:] = AT.hnorm(Amcc[i,:],normtype)
    Acdn[i,:] = AT.hnorm(Acd[i,:],normtype)
    Amccdn[i,:] = AT.hnorm(Amccd[i,:],normtype)

Acnd = np.zeros((nspec,npts),dtype=float)
Acnmc = meancenter(Acn)
Acnmcd = np.zeros((nspec,npts),dtype=float)
for i in range(nspec):
    Acnmcd[i,:] = AT.sg(Acnmc[i,:],25,2,deriv=2,rate=1)
    Acnd[i,:] = AT.sg(Acn[i,:],25,2,deriv=2,rate=1)

Acndmc = meancenter(Acnd)

## Dictionaries of All of the Preprocessed Data##

dslist = {'A':A,'Amc':Amc,'Ac':Ac,'Amcc':Amcc,'Acd':Acd,
          'Amccd':Amccd,'Acn':Acn,'Acnmc':Acnmc,'Amccn':
          Amccn,'Acdn':Acdn,'Amccdn':Amccdn,'Acnd':Acnd,
         'Acnmc':Acnmc,'Acnmcd':Acnmcd,'Acndmc':Acndmc}


labels = ['Aedes japonicus','Aedes albopictus','
Aedes triseriatus','Aedes aegypti']

labeldictN = {0:'Aedes japonicus',\
              1:'Aedes albopictus',\
              2:'Aedes triseriatus',\
              3:'Aedes aegypti'}

labeldictL = {'japonicus':'Aedes japonicus',\
              'albopictus':'Aedes albopictus',\
              'triseriatus':'Aedes triseriatus',\
              'aegypti':'Aedes aegypti'}

optidict = {0:'fp',\
            1:'fp',\
            2:'fp',\
            3:'fp'}

sexdict = {'m':1,'f':0,'1':1}
```

```
colordict = {"japonicus": 0, "albopictus": 1,
               "triseriatus": 2, "aegyptia": 3}

colordictR = {0:"japonicus",1: "albopictus",
               2:"triseriatus",3: "aegyptia"}

specdict = {"japonicus": 0, "albopictus": 1,
               "triseriatus": 2, "aegyptia": 3}

grpdict = {"japonicus_f": 0, "albopictus_f": 1,
            "triseriatus_f": 2, "aegyptia_f": 3,
           "japonicus_m": 4, "albopictus_m": 5,
            "triseriatus_m": 6, "aegyptia_m": 7}

#These dictionaries are used for making average spectra.

japdict = {"japonicus": 0, "albopictus": 1,
            "triseriatus": 1, "aegyptia": 1}

trisdict = {"japonicus": 1, "albopictus": 1,
            "triseriatus": 0, "aegyptia": 1}

albodict = {"japonicus": 1, "albopictus": 0,
            "triseriatus": 1, "aegyptia": 1}

aegdict =  {"japonicus": 1, "albopictus": 1,
            "triseriatus": 1, "aegyptia": 0}


###### One step classification
## this idea is to join the sex and species as a group
## Ex.  japonicus_f is one group and japonicus_m is another
##group
if sexflag == 'on':
    species = mdf['species'].unique()
    truearray = np.array([specdict[x] for x in mdf['species']])
    pass
else:
    mdf['grp'] = mdf[['species','sex']].apply(lambda
    x: '_'.join(x),
    axis=1)
    species = mdf['grp'].unique()
    truearray = np.array([grpdict[x] for x in mdf['grp']])

#
ngrps = len(np.unique(truearray))
```

Averaging Mosquito spectra by species

```
In [10]: japarray = np.array([japdict[x] for x in mdf['species']])
         trisarray = np.array([trisdict[x] for x in mdf['species']])
         albоarray = np.array([albodict[x] for x in mdf['species']])
         aegarray = np.array([aegdict[x] for x in mdf['species']])


         ## determine species mean and second derivative
         Acgrpmean = getgrpmean(Acn,truearray)
         Acgrpdmean = getgrpmean(Acnd,truearray)

         ## determine species standard deviation
         Acgrpstd = getgrpstd(Acn,truearray)

         ## determine species

         ##get mean, mean derivative, max, and min for each
       ##groups separately

         #Japonicus
         AcgrpmeanJ = getgrpmean(Acn,japarray)
         #Mean of normalized group
         AcgrpdmeanJ = getgrpmean(Acnd,japarray)
         #Mean of derivative group
         AcgrpmaxJ = getgrpmax(Acn,japarray)
         #Max of normalized group
         AcgrpminJ = getgrpmin(Acn,japarray)
         #Min of normalized group

         #Triseriatus
         AcgrpmeanT = getgrpmean(Acn,trisarray)
         AcgrpdmeanT = getgrpmean(Acnd,trisarray)
         AcgrpmaxT = getgrpmax(Acn, trisarray)
         AcgrpminT = getgrpmin(Acn,trisarray)

         #Albopictus
         AcgrpmeanAl = getgrpmean(Acn,alboarray)
         AcgrpdmeanAl = getgrpmean(Acnd,alboarray)
         AcgrpmaxAl = getgrpmax(Acn,alboarray)
         AcgrpminAl = getgrpmin(Acn,alboarray)

         #Aegypti
         AcgrpmeanAg = getgrpmean(Acn,aegarray)
         AcgrpdmeanAg = getgrpmean(Acnd,aegarray)
         AcgrpmaxAg = getgrpmax(Acn,aegarray)
         AcgrpminAg = getgrpmin(Acn,aegarray)
```

0.1 Band Ratio

- Band ratio is performed for data exploration.
- Distribution of ratio values can tell us some information on how certain structures are varying within and between mosquito species.

```
In [32]: ## creates a new dataframe of absorbance values at specified
           wavenumber
         dfratios = df[[1650,1565,1160,1115,1080,1030]]
         dfratios['species'] = mdf['species']
         dfr=dfratios.sort_values(by='species')

         print(x.shape,Acgrpmean.shape)
         from matplotlib import gridspec
         Aratio = dfr.as_matrix()
         xratio = np.array(dfr.columns.tolist())
         nspec1,npts1 = Aratio.shape
         trueratioarray = np.array([specdict[x] for x in dfr['species']])

         bn1= 0
         bn2 = 1
         bn3= 2
         bn4 = 3
         bn5= 4
         bn6 = 5

         B1 = np.zeros((nspec1,npts1),dtype=float)
         B2 = np.zeros((nspec1,npts1),dtype=float)
         B3 = np.zeros((nspec1,npts1),dtype=float)
         B4 = np.zeros((nspec1,npts1),dtype=float)
         B5 = np.zeros((nspec1,npts1),dtype=float)
         B6 = np.zeros((nspec1,npts1),dtype=float)


         r1 = np.zeros((nspec1,npts1),dtype=float)
         r2 = np.zeros((nspec1,npts1),dtype=float)
         r3 = np.zeros((nspec1,npts1),dtype=float)
         r4 = np.zeros((nspec1,npts1),dtype=float)
         r5 = np.zeros((nspec1,npts1),dtype=float)
         r6 = np.zeros((nspec1,npts1),dtype=float)
         r7 = np.zeros((nspec1,npts1),dtype=float)
         r8 = np.zeros((nspec1,npts1),dtype=float)

         #extacts the absorbance values at each wavenumber and stores
         #them in seperate arrays
         for i in range(nspec1):
             B1[i,:] = Aratio[i,bn1]
             B2[i,:] = Aratio[i,bn2]
```

```
    B3[i,:] = Aratio[i,bn3]
    B4[i,:] = Aratio[i,bn4]
    B5[i,:] = Aratio[i,bn5]
    B6[i,:] = Aratio[i,bn6]


#ratio calculation
r1 = B4/B2
r2 = B6/B5
r3 = B5/B1
r4 = B6/B1
r5 = B3/B2
r6 = B5/B2
r7 = B6/B2
r8 = B5/B2


ngrps = len(labels)
sampleidx = np.arange(nspec1)
colornames = getcolors()


fig,ax = plt.subplots(figsize=(9,7),dpi=100)
fig,ax1 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax2 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax3 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax4 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax5 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax6 = plt.subplots(figsize=(9,7),dpi=100)
fig,ax7 = plt.subplots(figsize=(9,7),dpi=100)


for grp in range(ngrps):        ## this selects which group
    b=trueratioarray == grp     ## b is the group
    ax.scatter(sampleidx[b],r1[b,grp],color=colornames[grp])
    ax1.scatter(sampleidx[b],r2[b,grp],color=colornames[grp])
    ax2.scatter(sampleidx[b],r3[b,grp],color=colornames[grp])
    ax3.scatter(sampleidx[b],r4[b,grp],color=colornames[grp])
    ax4.scatter(sampleidx[b],r5[b,grp],color=colornames[grp])
    ax5.scatter(sampleidx[b],r6[b,grp],color=colornames[grp])
    ax6.scatter(sampleidx[b],r7[b,grp],color=colornames[grp])
    ax7.scatter(sampleidx[b],r8[b,grp],color=colornames[grp])


ax.set_xlabel('Sample Number')
ax.set_ylabel('Bands Ratio')

ax1.set_xlabel('Sample Number')
```

```
ax1.set_ylabel('Bands Ratio')

ax2.set_xlabel('Sample Number')
ax2.set_ylabel('Bands Ratio')

ax3.set_xlabel('Sample Number')
ax3.set_ylabel('Bands Ratio')

ax4.set_xlabel('Sample Number')
ax4.set_ylabel('Bands Ratio')

ax5.set_xlabel('Sample Number')
ax5.set_ylabel('Bands Ratio')

ax6.set_xlabel('Sample Number')
ax6.set_ylabel('Bands Ratio')

ax7.set_xlabel('Sample Number')
ax7.set_ylabel('Bands Ratio')



ax.set_xlim((-10,500))
ax1.set_xlim((-10,500))
ax2.set_xlim((-10,500))
ax3.set_xlim((-10,500))
ax4.set_xlim((-10,500))
ax5.set_xlim((-10,500))
ax6.set_xlim((-10,500))
ax7.set_xlim((-10,500))
```

PLS-DA classification

- notes:
- instead of classifying spectra in to 4 classes, do 4 classifications into Positive or Negative classes
- can choose a threshold decision value that is very conservative and allows needs interventionby human expert
- look at notes to see what desirable TP,TN,FP,FN errors we should optimize on

classification functions

- this will allow for easier optimizaiton

```
In [14]: ## PLS-DA working this out using shuffle split
         ###set X preprocessing
         X = Acndmc
         #X=Amccd
```

```
### set threshold dict
threshdict = {0:0.5,\
              1:0.5,\
              2:0.5,\
              3:0.5}

## set nlv
nlv =10

labels = ['Aedes japonicus','Aedes albopictus','
Aedes triseriatus','Aedes aegypti']
from sklearn.model_selection import train_test_split,
cross_val_score,cross_val_predict,ShuffleSplit
from sklearn.metrics import classification_report

####this gives the same results as train_test_split
# and allows us to know which spectra are not working

Y = truearray
rs = ShuffleSplit(n_splits=1, test_size=0.50, random_state=42)
rs.get_n_splits(X)
for train_index, test_index in rs.split(X):
    Xtrain, Xtest = X[train_index], X[test_index]
    Ytrain, Ytest = Y[train_index], Y[test_index]

ngrps = len(labels)

#PLS
from sklearn.metrics import confusion_matrix
from sklearn.cross_decomposition import PLSRegression
##need to optimize nlv (don't overfit), need to mean
##center data
model = PLSRegression(n_components=nlv, scale=False)
nspec,npts = Xtest.shape
P = np.zeros((nspec,ngrps),dtype=float)
dfr = pd.DataFrame(columns=labels)# report
dfr1 = pd.DataFrame(columns=labels)# report

#prediction results
dfpr = pd.DataFrame(columns=labels,index=specidx[test_index])

for i in range(ngrps):
    ##makes boolean 0 or 1, 1 for group in question 0 for all
    other groups
    DAtrue = (Ytrain == i)*1.0
    model.fit(Xtrain,DAtrue)
    S = model.x_scores_
```

```
            P[:,i] = model.predict(Xtest)[:,0] ##prediction for test
            ##set

            T = Ytest == i
           ## if greater than threshold then assign true else False
            Pr = P[:,i] > threshdict[i]
            dfpr[labels[i]] = Pr

            #compares prediction to true answer
            tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()

            dfr1[labels[i]] = confusion_matrix(T, Pr).ravel()

            #classification performance parmameters are calculated
            Sr = calc_qual(T,tp,tn,fp,fn)
            dfr[labels[i]] = Sr[0]

        dfr

        #dfr.to_latex()  needs formating to work correctly
        #leaving this here to copy and paste into LaTeX
```

Optimization Functions

```
In [30]: def highlight_max(s):
             '''
             highlight the maximum in a Series yellow.
             '''
             is_max = s == s.max()
             return ['background-color: yellow' if v else '' for v
             in is_max]

         def highlight_min(s):
             '''
             highlight the minimum in a Series yellow.
             '''
             is_min = s == s.min()
             return ['background-color: orange' if v else '' for
             v in is_min]

         def plsda(Xtrain,Ytrain,Xtest,Ytest):
             from sklearn.cross_decomposition import PLSRegression
             model = PLSRegression(n_components=9, scale=False)
             P = np.zeros((nspec,ngrps),dtype=float)
             d = {}
             d = []
             for i in range(ngrps):
                 DAtrue = (Ytrain == i)*1.0  ##makes boolean 0 or 1
```

```
            model.fit(Xtrain,DAtrue)#
            S = model.x_scores_
            P[:,i] = model.predict(Xtest)[:,0]
            #pick a threshold that minimizes the number
            #of fp and fn
            #create an optimization loop for this
            thresh = 0.5
            T = Ytest == i
            Pr = P[:,i] > thresh
            tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()
            repdict = {'tp':tp,'tn':tn,'fp':fp,'fn':fn}
            d.append(repdict[optidict[i]])
        return d

    def splitit(X,Y):
        Y = truearray
        rs = ShuffleSplit(n_splits=1, test_size=0.50,
        random_state=42)
        rs.get_n_splits(X)
        for train_index, test_index in rs.split(X):
            Xtrain, Xtest = X[train_index], X[test_index]
            Ytrain, Ytest = Y[train_index], Y[test_index]
        return Xtrain,Ytrain,Xtest,Ytest


    d = plsda(Xtrain,Ytrain,Xtest,Ytest)
    df = pd.DataFrame({'junk':[]})
    df2 = pd.Series(d)
    df['key'] = df2
    df['key2'] = df2
```

PLS-DA plot

- Scatter plot of model generated prediction values for each sample.
- Four different models are generated for each species.
- The threshold value is indicated by the dashed line, samples above dashed line are classified as the species in question for that model.
- Blue dots are the true assigned class for that model, red are the false samples.

```
In [52]: ###### code for creating threshold predictions scatter plots
         nspec, ngrps = P.shape#### P is the prediction from PLS DA

         labels = ['Aedes japonicus','Aedes albopictus','
         Aedes triseriatus','Aedes aegypti']

         threshdict = {0:0.5,
                       1:0.5,
                       2:0.5,
```

```
            3:0.5}

ngrps = len(labels)
sampleidx = np.arange(nspec)

grpidx = Y[test_index]
figthscat, (axaj,axaa,axat,axag) = plt.subplots(figsize
=(12,8),nrows=1,ncols=4,sharex=False,sharey=True)

axgrps = [axaj,axaa,axat,axag]

for grp in range(ngrps):
    thresh = threshdict[grp]
    axgrp = axgrps[grp]
    ##this makes the threshold line
    axgrp.axhline(y=thresh, alpha=0.3, linestyle='--',
    color='k')
    ## this selects which group
    b = grpidx == grp            ## b is the group
    nb = grpidx != grp           ## nb is not the group
    axgrp.scatter(sampleidx[b],P[b,grp],color='b')
    axgrp.scatter(sampleidx[nb],P[nb,grp],color='r')


axaj.annotate('Sample ID',
            xy=(2.9, -0.105), xycoords='axes fraction',
            xytext=(-50, 0), textcoords='offset pixels',
            horizontalalignment='right',
            verticalalignment='bottom',
            fontsize=13)

axaj.annotate('(a)',
            xy=(.4, .95), xycoords='axes fraction',
            xytext=(-50, 0), textcoords='offset pixels',
            horizontalalignment='right',
            verticalalignment='bottom',
            fontsize=13)


axaa.annotate('(b)',
            xy=(0.4, .95), xycoords='axes fraction',
            xytext=(-50, 0), textcoords='offset pixels',
            horizontalalignment='right',
            verticalalignment='bottom',
            fontsize=13)


axat.annotate('(c)',
```

```
                    xy=(0.4, 0.95), xycoords='axes fraction',
                    xytext=(-50, 0), textcoords='offset pixels',
                    horizontalalignment='right',
                    verticalalignment='bottom',
                    fontsize=13)

        axag.annotate('(d)',
                    xy=(0.4, 0.95), xycoords='axes fraction',
                    xytext=(-50, 0), textcoords='offset pixels',
                    horizontalalignment='right',
                    verticalalignment='bottom',
                    fontsize=13)

        #axaa.set_xlabel('Sample ID')
        axaj.set_ylabel('Species Prediction')

        axaa.set_xlim((-10,250))
        axaj.set_xlim((-10,250))
        axat.set_xlim((-10,250))
        axag.set_xlim((-10,250))
        axaj.set_ylim((-1.5,2))

        figthscat.savefig(fighole+'PLSscatter.png')
```

Optimize Preprocessing for classificaiton

- search through preprocessing choices (chains?) and calc the TP,TN, FP,FN
- Find choices that min. FP (Check with BByrd, FN)
- aegypti shouldnt be in this area, so we don't want to miss any classifications
- highlight with highlight_function

```
In [35]: ##optimization of PreProcessing
         def plsdasingle(Xtrain,Ytrain,Xtest,Ytest,threshdict,nlv):
             from sklearn.metrics import confusion_matrix
             from sklearn.cross_decomposition import PLSRegression
             model = PLSRegression(n_components=9, scale=False)
             nspec,npts = Xtest.shape
             ngrps = len(threshdict)
             #print(ngrps)
             P = np.zeros((nspec,ngrps),dtype=float)
             d = []

             for grp in range(ngrps):
                 #print(grp)
                 thresh = threshdict[grp]
                 DAtrue = (Ytrain == grp)*1.0  ##makes boolean 0 or 1
                 model.fit(Xtrain,DAtrue)#
                 #S = model.x_scores_
```

```
            P[:,grp] = model.predict(Xtest)[:,0]
            #thresh = 0.5  #pick a threshold that minimizes the
            #number of fp and fn, create an optimization loop
            #for this
            T = Ytest == grp
            Pr = P[:,grp] > thresh

            tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()
            score =  ((tp+tn)/len(T)) *100  ##accuracy
            #repdict = {'tp':tp,'tn':tn,'fp':fp,'fn':fn}
            d.append(score)
            #d.append(repdict[optidict[grp]])

        return d
    labels = labeldictN.values()
    ngrps = len(labels)
    from sklearn.cross_decomposition import PLSRegression
    ####get from NLV opti
    model = PLSRegression(n_components=10, scale=True)

    scoresdf = pd.DataFrame({'junk':[]})
    #scores = {}
    for key,X in dslist.items():
        Xtrain,Ytrain,Xtest,Ytest = splitit(X,truearray)
        score = plsdasingle(Xtrain,Ytrain,Xtest,Ytest,threshdict,9)
        scoresdf[key] = pd.Series(score)

    scoresdf.drop('junk',1,inplace=True)
    scoresdf = scoresdf.T
    scoresdf = scoresdf.rename(columns=labeldictN)
    #scoresdf
    print('Predictions of Separate Validation using different
    Preprocessing treatments')
    print(optidict)
    #scoresdf
    scoresdf.style.apply(highlight_min).apply(highlight_max)
```

Optimize Number of Loading Vectors

```
In [36]: labels = labeldictN.values()
         ngrps = len(labels)
         X = Amccd

         #this choice of X is from the preprocesing optimization
         Xtrain,Ytrain,Xtest,Ytest = splitit(X,truearray)
         from sklearn.cross_decomposition import PLSRegression

         maxnlv = 25
```

```
scoresdf = pd.DataFrame({'junk':[]})
for nlv in range(1,maxnlv):
    dfr = pd.DataFrame(columns=range(1,ngrps))# report
    nspec,npts = Xtest.shape
    P = np.zeros((nspec,ngrps),dtype=float)
    model = PLSRegression(n_components=nlv, scale=False)
    for i in range(ngrps):
        DAtrue = (Ytrain == i)*1.0  ##makes boolean 0 or 1
        model.fit(Xtrain,DAtrue)#
        S = model.x_scores_
        P[:,i] = model.predict(Xtest)[:,0]
        thresh = 0.5
        T = Ytest == i
        Pr = P[:,i] > thresh
        tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()
        accuracy = (tp+tn)/len(T) *100
        repdict = {'tp':tp,'tn':tn,'fp':fp,'fn':fn}
        scoresdf.loc[nlv,i] = accuracy

scoresdf.drop('junk',1,inplace=True)

scoresdf1 = scoresdf.rename(columns=labeldictN)
print('Predictions of Separate Validation Optimizing on
the NLV')
print(optidict)
scoresdf1.style.apply(highlight_min).apply(highlight_max)
```

Optimization of threshold

```
In [33]: # optimize threshold
         labels = labeldictN.values()
         ngrps = len(labels)

         #this choice of X is from the preprocesing optimization
         Xtrain,Ytrain,Xtest,Ytest = splitit(Acdn,truearray)
         from sklearn.cross_decomposition import PLSRegression

         ##range of threshold values to loop through 0.005 step
        # at a time
         threshrange = np.arange(0.3,0.9,0.05)


         scoresdf = pd.DataFrame({'junk':[]})

         for thresh in threshrange:
             dfr = pd.DataFrame(columns=range(1,ngrps))# report
             nspec,npts = Xtest.shape
```

```
    P = np.zeros((nspec,ngrps),dtype=float)
    model = PLSRegression(n_components=10, scale=False)
    ###nlv comes from nlv opti
    for i in range(ngrps):
        DAtrue = (Ytrain == i)*1.0  ##makes boolean 0 or 1
        model.fit(Xtrain,DAtrue)#
        S = model.x_scores_
        P[:,i] = model.predict(Xtest)[:,0]
        #thresh = 0.5
        T = Ytest == i
        Pr = P[:,i] > thresh
        tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()
        accuracy = ((tp+tn)/len(T)) *100
        repdict = {'tp':tp,'tn':tn,'fp':fp,'fn':fn}
        scoresdf.loc[thresh,i] = accuracy

scoresdf = scoresdf.rename(columns=labeldictN)
print('Predictions of Separate Validation Optimizing on
the prediction threshold')
print(optidict)
scoresdf.style.apply(highlight_min).apply(highlight_max)

threshold_optidf = scoresdf## store for latter plotting

####figure for optimzation of threshold
fig, axthreshopti = plt.subplots(figsize=(12,9),nrows=1,
ncols=1,sharex=False)

x1 = scoresdf.index.tolist()
y0 = scoresdf[labeldictN[0]].tolist()
y1 = scoresdf[labeldictN[1]].tolist()
y2 = scoresdf[labeldictN[2]].tolist()
y3 = scoresdf[labeldictN[3]].tolist()

axthreshopti.plot(x1,y0)
axthreshopti.plot(x1,y1)
axthreshopti.plot(x1,y2)
axthreshopti.plot(x1,y3)
labels = labeldictN.values()
print(labels)
axthreshopti.legend(labels)
axthreshopti.set_xlabel('Prediction Threshold')
axthreshopti.set_ylabel('Accuracy (%)')
axthreshopti.set_ylim((80,105))
axthreshopti.set_xlim((0.3,0.9))
plt.axvline(x=0.5,linestyle='--',color='k')

fig.savefig(fighole+'ThresholdOpti.png')
```

PLS-DA Optimization of all perameters at once

- loops through several combinations of preprocessing, threshold and number of loading vectors

```
In [26]: def plsda_opti(Xtrain,Ytrain,Xtest,Ytest,thresh,nlv):
             '''
             This function takes data from only one group
             '''
             #from sklearn.metrics import confusion_matrix
             #from sklearn.cross_decomposition import PLSRegression
             model = PLSRegression(n_components=nlv, scale=False)
             nspec,npts = Xtest.shape

             #print(ngrps)
             P = np.zeros((nspec),dtype=float)
             d = []

             model.fit(Xtrain,Ytrain)#

             P = model.predict(Xtest)[:,0]
             T = Ytest
             Pr = P > thresh
             tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()
             score = (tp+tn)/len(T)    ##accuracy

             return score

         def optiloop(grp,threshrange,nlvrange,dslist,truearray):
             scores = []
             ppindx = [] ## stores preprocessing
             threshidx = [] ## stores threshold values
             nlvidx = []  ## stores number of loading vectors

             ##dslist is list of all preprocessing steps
             for key,X in dslist.items():
                 Xtrain,Ytrain,Xtest,Ytest = splitit(X,truearray)
                 Ytrain = (Ytrain == grp)*1.0
                 Ytest = (Ytest == grp)*1.0
                 for thresh in threshrange:
                     for nlv in nlvrange:
                         ppindx.append(key)
                         threshidx.append(thresh)
                         nlvidx.append(nlv)
                         scores.append(plsda_opti(Xtrain,Ytrain,
                         Xtest,Ytest,thresh,nlv))
             tuples = list(zip(ppindx,threshidx,nlvidx))
             index = pd.MultiIndex.from_tuples(tuples, names=['PP',
             'thresh','nlv'])
```

```
    S = pd.Series(scores, index=index)
    return S


from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import confusion_matrix

labels = ['Aedes japonicus','Aedes albopictus','
Aedes triseriatus','Aedes aegypti']

threshrange = np.arange(0.1,0.9,0.05)
nlvrange = range(7,20)
# dslist  is the list of preprocessed data sets
scores = []
ppindx = []
threshidx = []
nlvidx = []

serieslist = [ ]

for grp,label in enumerate(labels):
    S = optiloop(grp,threshrange,nlvrange,dslist,truearray)
    serieslist.append(S)

dfs = pd.concat(serieslist,axis=1).rename(columns=labeldictN)

dfs

dfs.style.apply(highlight_min).apply(highlight_max)
```

Search for outliers

```
In [707]: ## more optimiation funcitons
          def fitpredict(Xtrain,Xtest,Ytrain,Ytest):
              nspec,npts = Xtest.shape
              P = np.zeros((nspec,ngrps),dtype=float)
              dfr = pd.DataFrame(columns=labels)# report
              #prediction results
              dfpr = pd.DataFrame(columns=labels,index=specidx
              [test_index])
              for i in range(ngrps):

                  DAtrue = (Ytrain == i)*1.0  ##makes boolean 0 or 1
                  model.fit(Xtrain,DAtrue)#
                  S = model.x_scores_
                  P[:,i] = model.predict(Xtest)[:,0]

                  T = Ytest == i
```

```
                Pr = P[:,i] > threshdict[i]
                dfpr[labels[i]] = Pr

                tn, fp, fn, tp = confusion_matrix(T, Pr).ravel()


                Sr = calc_qual(T,tp,tn,fp,fn)
                dfr[labels[i]] = Sr[0]
            return dfr.T.mc
```

In [54]:
```
        ##search for ouliers
        ###set X preprocessing
        X = Acd
        ### set threshold dict
        threshdict = {0:0.5,\
                      1:0.5,\
                      2:0.5,\
                      3:0.5}

        ## set nlv
        nlv = 13
        labels = ['Aedes japonicus','Aedes albopictus','
        Aedes triseriatus','Aedes aegypti']
        from sklearn.model_selection import train_test_split,
         cross_val_score, cross_val_predict,ShuffleSplit
        from sklearn.metrics import classification_report

        ngrps = len(labels)
        #PLS
        from sklearn.metrics import confusion_matrix
        from sklearn.cross_decomposition import PLSRegression

        ##need to optimize nlv (don't overfit), need to mean
         ##center data
        model = PLSRegression(n_components=nlv, scale=False)


        scoresdf = pd.DataFrame({'junk':[]})
        ####this gives the same results as train_test_split
        # and allows us to know which spectra are not working
        Y = truearray
        rs = ShuffleSplit(n_splits=100, test_size=0.20)
        #, random_state=42)
        rs.get_n_splits(X)
        i = 0
        traindict = {}
        testdict = {}
        for train_index, test_index in rs.split(X):
```

```
            traindict[i] = train_index
            testdict[i] = test_index
            Xtrain, Xtest = X[train_index], X[test_index]
            Ytrain, Ytest = Y[train_index], Y[test_index]
            scoresdf[i] = fitpredict(Xtrain,Xtest,Ytrain,Ytest)
            i = i + 1

        scoresdf.drop('junk',1,inplace=True)
        scoresdf.T.style.apply(highlight_min).apply(highlight_max)
```

Begin figure output

Example of IR spectrum of Mosquito

```
In [24]: sampleA = Acn[0,:]
         samplex = x
         fig, axexample = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)

         #All groups example spectra plotted
         #axexample.plot(x,Acgrpmean.T)

         #One group example spectra plotted
         axexample.plot(samplex,sampleA, color = 'black')

         axexample.set_xlim((650,3800))
         axexample.invert_xaxis()


         labels = ['Aedes japonicus','Aedes albopictus','
         Aedes triseriatus','Aedes aegypti']
         axexample.legend(labels)

         ##vertical lines
         plt.axvline(x=1640,linestyle='--',color='r')

         axexample.set_xlabel('Wavenumber $\mathregular{cm^-1}$')
         axexample.set_ylabel('Absorbance')
         fig.savefig(fighole+'examplespec.png')
```

Plot mean spectra for each species

```
In [39]: print(x.shape,Acgrpmean.shape)

         fig, axmeanspec = plt.subplots(figsize=(12,9),nrows=1,
         ncols=1,
         sharex=False)
         axmeanspec.plot(x,Acgrpmean.T)
```

```
axmeanspec.set_xlim((650,1800))
axmeanspec.invert_xaxis()


labels = species
labels = labeldictN.values()
print(labels)
axmeanspec.legend(labels)

##vertical lines

plt.axvline(x=1550,linestyle='--',color='k')
plt.axvline(x=1030,linestyle='--',color='k')
plt.axvline(x=1160,linestyle='--',color='k')
plt.axvline(x=1241,linestyle='--',color='k')
plt.axvline(x=1115,linestyle='--',color='k')
plt.axvline(x=1080,linestyle='--',color='k')


axmeanspec.set_xlabel('Wavenumber /$cm^{-1}$')
axmeanspec.set_ylabel('Normalized Absorbance')
fig.savefig(fighole+'meanspec1.png')
```

Plot Second Derivative Spectra

```
In [48]: print(x.shape,Acgrpdmean.shape)
         print(x.shape,Acgrpmean.shape)

         labels = ['Aedes japonicus','Aedes triseriatus','
         Aedes albopictus','Aedes aegypti']

         fig, axsdpec = plt.subplots(figsize=(12,9),nrows=1, ncols=1
         ,sharex=False)
         axsdpec.plot(x,Acgrpdmean.T)


         axsdpec.set_xlim((1830,710))
         axsdpec.set_ylim((-0.0012,0.0012))
         axsdpec.set_ylabel('Normalized Absorbance')


         axsdpec.legend(labels)
```

Plot standard deviation spectra for each species

```
In [34]: fig, axstdspec = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)
```

```
axstdspec.plot(x,Acgrpstd.T)
axstdspec.set_xlim((650,1800))
axstdspec.invert_xaxis()

labels = species
labels = labeldictN.values()
axstdspec.legend(labels)

axstdspec.set_xlabel('Wavenumber /$cm^{-1}$')
fig.savefig(fighole+'stdpec1.png')
```

Graphs of max, min and mean spectra for each species

```
In [53]: print(x.shape,AcgrpmeanJ.shape)
         #print(Acn)
         fig, axspec = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)
         fig, axspec1 = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)
         fig, axspec2 = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)
         fig, axspec3 = plt.subplots(figsize=(12,9),nrows=1, ncols=1,
         sharex=False)


         axspec.plot(x,AcgrpmeanJ[0].T)
         axspec.plot(x,AcgrpmaxJ[0].T)
         axspec.plot(x,AcgrpminJ[0].T)

         axspec1.plot(x,AcgrpmeanT[0].T)
         axspec1.plot(x,AcgrpmaxT[0].T)
         axspec1.plot(x,AcgrpminT[0].T)

         axspec2.plot(x,AcgrpmeanAl[0].T)
         axspec2.plot(x,AcgrpmaxAl[0].T)
         axspec2.plot(x,AcgrpminAl[0].T)

         axspec3.plot(x,AcgrpmeanAg[0].T)
         axspec3.plot(x,AcgrpmaxAg[0].T)
         axspec3.plot(x,AcgrpminAg[0].T)

         axspec.set_xlim((650,1800))
         axspec.invert_xaxis()
         axspec1.set_xlim((650,1800))
         axspec1.invert_xaxis()
         axspec2.set_xlim((650,1800))
         axspec2.invert_xaxis()
         axspec3.set_xlim((650,1800))
```

```
axspec3.invert_xaxis()

axspec.set_xlabel('Wavenumber /$cm^{-1}$')
axspec.set_ylabel('Normalized Absorbance')
axspec1.set_xlabel('Wavenumber /$cm^{-1}$')
axspec1.set_ylabel('Normalized Absorbance')
axspec2.set_xlabel('Wavenumber /$cm^{-1}$')
axspec2.set_ylabel('Normalized Absorbance')
axspec3.set_xlabel('Wavenumber /$cm^{-1}$')
axspec3.set_ylabel('Normalized Absorbance')

axspec.set_title('$Ae. japonicus$')
axspec1.set_title('$Ae. triseriatus$')
axspec2.set_title('$Ae. albopictus$')
axspec3.set_title('$Ae. aegypti$')


#fig.savefig(fighole+'meanspec1.png')
```

Other Plots

PLS 3D Analysis

```
In [14]: from mpl_toolkits.mplot3d import axes3d
         import matplotlib.pyplot as plt

         fig = plt.figure(figsize=(8,6))
         axpca3d = fig.add_subplot(111, projection='3d')

         colornames1 = getcolors()

         for i in range(ngrps):
             for idx in [0,1,2,3,4,5,6,7]:
                 b = Ytrain == i
                 DAtrue = (Ytrain == i)*1.0  ##makes boolean 0 or 1
                 model.fit(Xtrain,DAtrue)#
                 S = model.x_scores_
                 b = Ytrain == idx
                 axpca3d.scatter(S[b,1],S[b,2], color=colornames1[i],
                  alpha = 1, lw=1)

         labels = ['Aedes japonicus','Aedes albopictus','
         Aedes triseriatus','Aedes aegypti']
```